DOCUMENT RESUME

ED 048 910 LI 002 719

TITLE	Information Storage and Retrieval...Reports on Analysis, Dictionary Construction, User Feedback, Clustering, and On-Line Retrieval.
INSTITUTION	Cornell Univ., Ithaca, N.Y. Dept. of Computer Science.
SPONS AGENCY	National Library of Medicine (DHEW), Bethesda, Md.; National Science Foundation, Washington, D.C.
REPORT NO	ISR-18
PUB DATE	Oct 70
NOTE	523p.
EDRS PRICE	EDRS Price MF-$0.65 HC-$19.74
DESCRIPTORS	Automatic Indexing, Automation, Classification, Content Analysis, *Dictionaries, Electronic Data Processing, *Feedback, *Information Retrieval, *Information Storage, Relevance (Information Retrieval), Thesauri, *Use Studies
IDENTIFIERS	On Line Retrieval Systems, *Saltons Magical Automatic Retriever of Texts, SMART

ABSTRACT

Part One, Automatic Content Analysis contains: "Content Analysis in Information Retrieval;" "The 'Generality' Effect and the Retrieval Evaluation for Large Collections;" "Automatic Indexing Using Bibliographic Citations" and "Automatic Resolution of Ambiguities from Natural Language Text." Part Two, Automatic Dictionary Construction, includes: "The Effect of Common Words and Synonyms on Retrieval Performance;" "Negative Dictionaries" and "Experiments in Automatic Thesaurus Construction for Information Retrieval." Part Three, user feedback procedures, includes: "Variations on the Query Splitting Technique with Relevance Feedback;" "Effectiveness of Feedback Strategies on Collections of Differing Generality;" "Selective Negative Feedback Methods" and "The Use of Past Relevance Decisions in Relevance Feedback." Part Four, clustering methods, includes: "A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering" and "A Systematic Study of Query-Clustering Techniques: A Progress Report." Part Five, on-line retrieval system design, contains: "A Prototype On-Line Document Retrieval System" and "Template Analysis in a Conversational System." (Author:NH)

"PERMISSION TO REPRODUCE THIS COPYRIGHTED MATERIAL HAS BEEN GRANTED BY
Dept. of Computer Science, Cornell Univ.
TO ERIC AND ORGANIZATIONS OPERATING UNDER AGREEMENTS WITH THE U.S. OFFICE OF EDUCATION. FURTHER REPRODUCTION OUTSIDE THE ERIC SYSTEM REQUIRES PERMISSION OF THE COPYRIGHT OWNER."

ED048910

Department of Computer Science

Cornell University

Ithaca, New York 14850

Scientific Report No. ISR-18

INFORMATION STORAGE AND RETRIEVAL

to

The National Science Foundation

and to

The National Library of Medicine

Reports on Analysis, Dictionary Construction, User Feedback, Clustering, and On-Line Retrieval

LI 002 719

Ithaca, New York

October 1970

Gerard Salton

Project Director

©

Copyright, 1970
by Cornell University

Use, reproduction, or publication, in whole or in part, is permitted for any purpose of the United States Government.

2

SMART Project Staff

Robert Crawford
Barbara Galaska
Eileen Gudat
Marcia Kerchner
Ellen Lundell
Robert Peck
Jacob Razon
Gerard Salton
Donna Williamson
Robert Williamson
Steven Worona
Joel Zumoff

TABLE OF CONTENTS

Page

SUMMARY xv

PART ONE

AUTOMATIC CONTENT ANALYSIS

I. WEISS, S. F.
"Content Analysis in Information Retrieval"

Abstract I-1

1. Introduction I-2

2. ADI Experiments I-5

   A) Statistical Phrases I-5
   B) Syntactic Phrases I-7
   C) Cooccurrence. I-9
   D) Elimination of Phrase List I-12
   E) Analysis of ADI Results I-20

3. The Cranfield Collection I-26

4. The TIME Subset Collection. I-27
   A) Construction. I-27
   B) Analysis of Results I-31

5. A Third Collection I-39

6. Conclusion I-43

References I-46

II. SALTON, G.
"The 'Generality' Effect and the Retrieval Evaluation for Large Collections"

4

TABLE OF CONTENTS (continued)

Page

II. continued

Abstract II-1

1. Introduction. II-1

2. Basic System Parameters II-3

3. Variations in Collection Size II-7

A) Theoretical Considerations II-7

B) Evaluation Results II-10

C) Feedback Performance II-15

4. Variations in Relevance Judgments. II-24

5. Summary II-31

References. II-33

III. SALTON, G.
"Automatic Indexing Using Bibliographic Citations"

Abstract III-1

1. Significance of Bibliographic Citations. III-1

2. The Citation Test III-4

3. Evaluation Results. III-9

References. III-19

Appendix III-20

IV. WEISS, S. F.
"Automatic Resolution of Ambiguities from Natural Language Text"

TABLE OF CONTENTS (continued)

Page

IV. continued

Abstract IV-1

1. Introduction. IV-2

2. The Nature of Ambiguities IV-4

3. Approaches to Disambiguation IV-8

4. Automatic Disambiguation. IV-14

   A) Application of Extended Template Analysis to Disambiguation IV-14

   B) The Disambiguation Process IV-15

   C) Experiments IV-17

   D) Further Disambiguation Processes IV-20

5. Learning to Disambiguate Automatically IV-21

   A) Introduction IV-21

   B) Dictionary and Corpus IV-21

   C) The Learning Process IV-23

   D) Spurious Rules IV-28

   E) Experiments and Results IV-30

   F) Extensions. IV-46

6. Conclusion IV-49

References. IV-50

PART TWO

AUTOMATIC DICTIONARY CONSTRUCTION

V. BERGMARK, D.

TABLE OF CONTENTS (continued)

Page

V. continued

"The Effect of Common Words and Synonyms on Retrieval Performance"

Abstract V-1

1. Introduction. V-1

2. Experiment Outline. V-2

   A) The Experimental Data Base V-2
   B) Creation of the Significant Stem Dictionary. . . . V-2
   C) Generation of New Query and Document Vectors . . . V-4
   D) Document Analysis — Search and Average Runs. . . . V-5

3. Retrieval Performance Results V-7

   A) Significant vs. Standard Stem Dictionary. V-7
   B) Significant Stem vs. Thesaurus V-9
   C) Standard Stem vs. Thesaurus V-11
   D) Recall Results V-11
   E) Effect of "Query Wordiness" on Search Performance. . V-15
   F) Effect of Query Length on Search Performance . . . V-15
   G) Effect of Query Generality on Search Performance . . V-17
   H) Conclusions of the Global Analysis. V-19

4. Analysis of Search Performance. V-20

5. Conclusions V-31

6. Further Studies. V-32

References. V-34

Appendix I. V-35

Appendix II V-39

TABLE OF CONTENTS (continued)

Page

VI. BONWIT, K. and ASTE-TONSMANN, J.
"Negative Dictionaries"

Abstract VI-1

1. Introduction. VI-1

2. Theory. VI-2

3. Experimental Results VI-7

4. Experimental Method VI-19

   A) Calculating Q_i VI-19

   B) Deleting and Searching. VI-20

5. Cost Analysis VI-25

6. Conclusions VI-29

References. VI-33

VII. SALTON, G.
"Experiments in Automatic Thesaurus Construction for Information Retrieval"

Abstract VII-1

1. Manual Dictionary Construction. VII-1

2. Common Word Recognition VII-8

3. Automatic Concept Grouping Procedures VII-17

4. Summary VII-25

References. VII-26

8

TABLE OF CONTENTS (continued)

Page

PART THREE

USER FEEDBACK PROCEDURES

VIII. BAKER, T. P.
"Variations on the Query Splitting Technique with Relevance Feedback"

Abstract VIII-1

1. Introduction. VIII-1

2. Algorithms for Query Splitting. VIII-3

3. Results of Experimental Runs VIII-11

4. Evaluation VIII-23

References. VIII-25

IX. CAPPS, B. and YIN, M.
"Effectiveness of Feedback Strategies on Collections of Differing Generality"

Abstract IX-1

1. Introduction. IX-1

2. Experimental Environment. IX-3

3. Experimental Results IX-8

4. Conclusion IX-19

References. IX-23

Appendix IX-24

9

TABLE OF CONTENTS (continued)

Page

X. KERCHNER, M.
"Selective Negative Feedback Methods"

Abstract X-1
1. Introduction. X-1
2. Methodology X-2
3. Selective Negative Relevance Feedback Strategies. . . X-5
4. The Experimental Environment X-6
5. Experimental Results X-8
6. Evaluation of Experimental Results X-13
References. X-20

XI. PAAVOLA, L.
"The Use of Past Relevance Decisions in Relevance Feedback"

Abstract XI-1
1. Introduction. XI-1
2. Assumptions and Hypotheses XI-2
3. Experimental Method XI-3
4. Evaluation XI-7
5. Conclusion XI-12
References. XI-14

TABLE OF CONTENTS (continued)

Page

PART FOUR

CLUSTERING METHODS

XII. JOHNSON, D. B. and LAFUENTE, J. M.
"A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering"

Abstract XII-1

1. Introduction. XII-1

2. Methods of Clustering. XII-3

3. Strategy XII-5

4. The Algorithm XII-6

    A) Cluster Size XII-8
    B) Number of Clusters XII-9
    C) Overlap. XII-10
    D) An Example. XII-10

5. Implementation XII-13

    A) Storage Management XII-14

6. Results XII-14

    A) Clustering Costs. XII-15
    B) Effect of Document Ordering XII-19
    C) Search Results on Clustered ADI Collection . . . XII-20
    D) Search Results of Clustered Cranfield Collection . XII-31

7. Conslusions XII-34

References. XII-37

TABLE OF CONTENTS (continued)

Page

XIII. WORONA, S.
"A Systematic Study of Query-Clustering Techniques: A Progress Report"

Abstract XIII-1

1. Introduction. XIII-1

2. The Experiment XIII-4

    A) Splitting the Collection XIII-4
    B) Phase 1: Clustering the Queries XIII-6
    C) Phase 2: Clustering the Documents. XIII-8
    D) Phase 3: Assigning Centroids XIII-12
    E) Summary. XIII-13

3. Results XIII-13

4. Principles of Evaluation. XIII-16

References. XIII-22

Appendix A. XIII-24

Appendix B. XIII-29

Appendix C. XIII-36

PART FIVE

ON-LINE RETRIEVAL SYSTEM DESIGN

XIV. WILLIAMSON, D. and WILLIAMSON, R.
"A Prototype On-Line Document Retrieval System"

Abstract XIV-1

TABLE OF CONTENTS (continued)

Page

XIV. continued

1. Introduction. XIV-1
2. Anticipated Computer Configuration XIV-2
3. On-Line Document Retrieval — A User's View. XIV-4
4. Console Driven Document Retrieval — An Internal View . XIV-10
   A) The Internal Structure. XIV-10
   B) General Characteristics of SMART Routines . . . XIV-16
   C) Pseudo-Batching XIV-17
   D) Attaching Consoles to SMART XIV-19
   E) Console Handling — The Superivsor Interface . . XIV-21
   F) Parameter Vectors XIV-21
   G) The Flow of Control. XIV-22
   H) Timing Considerations XIV-23
   I) Noncore Resident Files. XIV-26
   J) Core Resident Files. XIV-28
5. Consol — A Detailed Look. XIV-30
   A) Competition for Core XIV-30
   B) The SMART On-line Console Control Block XIV-31
   C) The READY Flag and the TRT Instruction XIV-32
   D) The Routines LATCH, CONSIN, and CONSOT XIV-32
   E) CONSOL as a Traffic Controller XIV-34
   F) A Detailed View of CYCLE XIV-37
6. Summary XIV-39

Appendix XIV-40

XV. WEISS, S. F.
"Template Analysis in a Conversational System"

TABLE OF CONTENTS (continued)

Page

XV. continued

Abstract XV-1

1. Motivation XV-1

2. Some Existing Conversational Systems. XV-4

3. Goals for a Proposed Conversational System. . . XV-7

4. Implementation of the Conventional System . . . XV-11

    A) Capabilities XV-11
    B) Input Conventions XV-12
    C) The Structure of the Process. XV-13
    D) Template Analysis in the Conversational System XV-14
    E) The Guide Facility XV-23
    F) Tutorials XV-24

5. Experimentation. XV-25

    A) System Performance XV-30
    B) User Performance. XV-31
    C) Timing XV-34

6. Future Extensions XV-35

7. Conclusion XV-37

References. XV-39

Summary

The present report is the eighteenth in a series describing research in automatic information storage and retrieval conducted by the Department of Computer Science at Cornell University. The report covering work carried out by the SMART project for approximately one year (summer 1969 to summer 1970) is separated into five parts: automatic content analysis (Sections I to IV), automatic dictionary construction (Sections V to VII), user feedback procedures (Sections VIII to XI), document and query clustering methods (Sections XII and XIII), and SMART systems design for on-line operations (Sections XIV and XV).

Most recipients of SMART project reports will experience a gap in the series of scientific reports received to date. Report ISR-17, consisting of a master's thesis by Thomas Brauen entitled "Document Vector Modification in On-line Information Retrieval Systems" was prepared for limited distribution during the fall of 1969. Report ISR-17 is available from the National Technical Information Service in Springfield, Virginia 22151, under order number PB 186-135.

The SMART system continues to operate in a batch processing mode on the IBM 360 model 65 system at Cornell University. The standard processing mode is eventually to be replaced by an on-line system using time-shared console devices for input and output. The overall design for such an on-line version of SMART has been completed, and is described in Section XIV of the present report. While awaiting the time-sharing implementation of the system, new retrieval experiments have been performed using larger document collections within the existing system. Attempts to compare the performance

of several collections of different sizes must take into account the collection "generality". A study of this problem is made in Section II of the present report. Of special interest may also be the new procedures for the automatic recognition of "common" words in English texts (Section VI), and the automatic construction of thesauruses and dictionaries for use in an automatic language analysis system (Section VII). Finally, a new inexpensive method of document classification and term grouping is described and evaluated in Section XII of the present report.

Sections I to IV cover experiments in automatic content analysis and automatic indexing. Section I by S. F. Weiss contains the results of experiments, using statistical and syntactic procedures for the automatic recognition of phrases in written texts. It is shown once again that because of the relative heterogeneity of most document collections, and the sparseness of the document space, phrases are not normally needed for content identification.

In Section II by G. Salton, the "generality" problem is examined which arises when two or more distinct collections are compared in a retrieval environment. It is shown that proportionately fewer nonrelevant items tend to be retrieved when larger collections (of low generality) are used, than when small, high generality collections serve for evaluation purposes. The systems viewpoint thus normally favors the larger, low generality output, whereas the user viewpoint prefers the performance of the smaller collection.

The effectiveness of bibliographic citations for content analysis purposes is examined in Section III by G. Salton. It is shown that in some situations when the citation space is reasonably dense, the use of citations attached to documents is even more effective than the use of standard keywords or descriptors. In any case, citations should be added to the normal descriptors whenever they happen to be available.

In the last section of Part 1, certain template analysis methods are applied to the automatic resolution of ambiguous constructions (Section IV by S. F. Weiss). It is shown that a set of contextual rules can be constructed by a semi-automatic learning process, which will eventually lead to an automatic recognition of over ninety percent of the existing textual ambiguities.

Part 2, consisting of Sections V, VI and VII covers procedures for the automatic construction of dictionaries and thesauruses useful in text analysis systems. In Section V by D. Bergmark it is shown that word stem methods using large common word lists are more effective in an information retrieval environment that some manually constructed thesauruses, even though the latter also include synonym recognition facilities.

A new model for the automatic determination of "common" words (which are not to be used for content identification) is proposed and evaluated in Section VI by K. Bonwit and J. Aste-Tonsmann. The resulting process can be incorporated into fully automatic dictionary construction systems. The complete thesaurus construction problem is reviewed in Section VII by G. Salton, and the effectiveness of a variety of automatic dictionaries is evaluated.

Part 3, consisting of Sections VIII through XI, deals with a number of refinements of the normal relevance feedback process which has been examined in a number of previous reports in this series. In Section VIII by T. P. Baker, a query splitting process is evaluated in which input queries are split into two or more parts during feedback whenever the relevant documents identified by the user are separated by one or more nonrelevant ones.

The effectiveness of relevance feedback techniques in an environment of variable generality is examined in Section IX by B. Capps and M. Yin. It is shown that some of the feedback techniques are equally applicable to collections of small and large generality. Techniques of negative feedback (when no relevant items are identified by the users, but only nonrelevant ones) are considered in Section X by M. Kerchner. It is shown that a number of selective negative techniques, in which only certain specific concepts are actually modified during the feedback process, bring good improvements in retrieval effectiveness over the standard nonselective methods.

Finally, a new feedback methodology in which a number of documents jointly identified as relevant to earlier queries are used as a set for relevance feedback purposes is proposed and evaluated in Section XI by L. Paavola.

Two new clustering techniques are examined in Part 3 of this report, consisting of Sections XII and XIII. A controlled, inexpensive, single-pass clustering algorithm is described and evaluated in Section XII by D. B. Johnson and J. M. Lafuente. In this clustering method, each document is examined only once, and the procedure is shown to be equivalent in certain circumstances to other more demanding clustering procedures.

The query clustering process, in which query groups are used to define the information search strategy is studied in Section XIII by S. Worona. A variety of parameter values is evaluated in a retrieval environment to be used for cluster generation, centroid definition, and final search strategy.

The last part, number five, consisting of Sections XIV and XV, covers the design of on-line information retrieval systems. A new SMART system design for on-line use is proposed in Section XIV by D. and R. Williamson, based on the concepts of pseudo-batching and the interaction of a cycling program with a console monitor. The user interface and conversational facilities are also described.

A template analysis technique is used in Section XV by S. F. Weiss for the implementation of conversational retrieval systems used in a time-sharing environment. The effectiveness of the method is discussed, as well as its implementation in a retrieval situation.

Additional automatic content analysis and search procedures used with the SMART system are described in several previous reports in this series, including notably reports ISR-11 to ISR-16 published between 1966 and 1969. These reports are all available from the National Technical Information Service in Springfield, Virginia.

G. Salton

I. Content Analysis in Information Retrieval

S. F. Weiss

Abstract

In information retrieval there exist a number of content analysis schemes which analyze natural language text to varying degrees of complexity. Regardless of how well the text analysis is performed by each process, the true value of a given process lies in its effectiveness as an information retrieval tool. The performance may in each case be investigated by actual retrieval tests using the various proposed content analysis schemes.

Results obtained with a variety of linguistic phrase recognition methods show that very little, if any, improvements in retrieval effectiveness are obtained when any of the refined content analysis schemes are used with existing document collections. The main reason appears to be the fact that the value of refined content analysis systems resides in their effectiveness in separating lexically similar, but semantically different documents. Existing collections are too sparse, and do not contain many close documents. When denser collections are created, it can be shown that linguistic content analysis methods become of increasing value as the density increases. The queries also influence the type of content analysis to be used. In general, queries of the question-answering variety show improved retrieval results with increasing refinements in the content analysis. Document retrieval queries do not exhibit this type of improvement.

Future work must be devoted to a determination of what makes a user judge a particular document to be relevant. With more insight into the relevance area, the role of linguistic content analysis in information retrieval may become more clearly defined.

1. Introduction

The purpose of a content analysis system as considered in this study is as an information retrieval aid. It is therefore necessary to perform retrieval using various content analysis methods to determine how well it fulfills its actual role. This study presents experiments and results aimed at determining the conditions under which content analysis improves retrieval results as well as the degree of improvement obtained. All information retrieval systems use some degree of content analysis in its broadest sense. This is generally in the form of assignment of concept indicators to individual words. But in this study content analysis refers to the analysis and utilization of multi-word groups as information retrieval tools.

Using phrases determined by content analysis as an information retrieval aid is theoretically very appealing. It adds another dimension to search capabilities beyond the single word matching used by most infromation retrieval systems. Documents and queries are matched not only on content, but on the interrelationship of content elements as well. Hutchins [3] has proposed an information retrieval system based solely on the cooccurrence of phrases in documents and queries. However, some experiments indicate that phrases alone may be too strict a criterion for useful results. A more reasonable approach is to use phrases in conjunction with a less structured method such as word or concept matching. Therefore in this study phrases are considered as an adjunct to single concept matching.

A number of existing information retrieval systems permit searching on multi-word structured information. Some systems such as that designed by Curtice and Jones at Arthur D. Little [1] index documents and queries by contiguous word pairs as well as individual words. Retrieval is thus aided by this rudimentary form of phrase analysis. The IBM Document Processing System [4] takes this capability one step further. Multi-word search keys can be specified using a number of options besides simple contiguity. For example, consider the sample queries below. Query A retrieves documents containing "information" and "retrieval" in that order and separated by at most one other word. Query B retrieves documents with the same two words separated by at most one word but with no restriction on ordering. This will retrieve "information retrieval" as well as "retrieval of information". Queries C and D further relax the proximity criterion and retrieve documents in which "information" and "retrieval" occur within the same sentence and the same paragraph respectively.

A. INFORMATION RETRIEVAL (+1)

B. INFORMATION RETRIEVAL (-+1)

C. INFORMATION RETRIEVAL (SEN)

D. INFORMATION RETRIEVAL (PAR)

This specification is an attempt to perform some degree of semantic normalization. It permits the association of phrases which are semantically similar but structurally different. However the IBM system and others like it approach the semantic normalization by structural rather than semantic means. The resultant semantic processes are hence necessarily very superficial. As Lesk points out, phrases determined by processes of this type may cooccur in documents and queries too infrequently for them to be of any practical value. Lesk therefore proposes an information retrieval system in which documents and queries are subjected to a complex syntactic and semantic analysis. Phrase normalization is then based on meaning rather than just structure [5]. A few other semantically based content analysis schemes exist such as the manual indexing process developed by Mandersloot, Douglas and Spicer [2]. Of all existing information retrieval systems with content analysis capabilities, the SMART system provides the greatest variety of content analysis methods. This makes SMART an excellent experimental facility for testing content analysis in general. The various SMART content analysis methods are presented in some detail later in this study.

In information retrieval, phrases can do two things. First, they can distinguish between two documents with similar content elements but different meaning. For example, the two inputs below are assigned identical concept vectors by normal text cracking methods. To distinguish between them requires that the structure as well as the content of the input be considered.

A. Design of computer systems
B. Computerized design systems

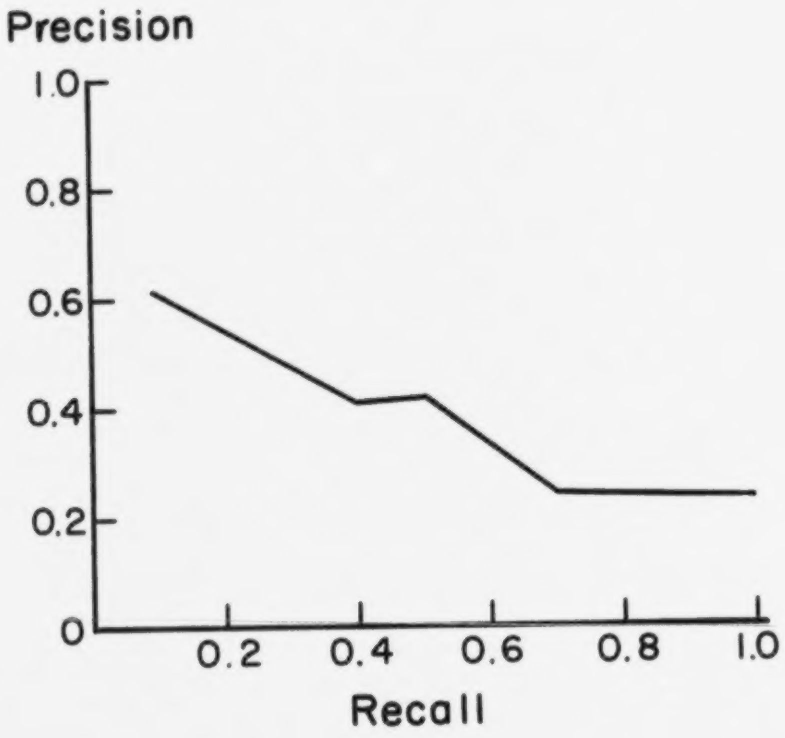
A second job performed by phrases is that of reinforcing correlations between queries and documents which have similar phrases. In this way the cooccurrence in the document and query of concepts which form a phrase is weighted more heavily than the cooccurrence of a similar number of unrelated concepts. While this might appear to be a convincing case in favor of using phrases in information retrieval, the previous argument is purely theoretical. It remains to test the theory by performing retrieval using various phrase determination methods. It is necessary to analyze the results obtained not only to determine how the overall results compare with those achieved without the use of phrases, but also to determine the exact cause of the phrase method results. That is, are the new results a function of the document or query collections used, the phrase determining technique, the matching procedure, or a combination of several factors?

2. ADI Experiments

The first set of experiments uses the ADI collection. This is a set of eighty-two documents and thirty-five queries in the field of documentation. About half of the queries ask for specific information while the other half are of a more general nature. A set of ten queries, five general and five specific, is chosen as representative of the various query forms and constructions. A normal SMART retrieval run is then performed on the entire ADI collection and the ten test queries. For each query the ten most highly correlated documents are identified. These documents along with any others, relevant to the test queries but not in the top ten, are collected to form a test document set. The total set contains 56 of the 82 ADI documents. In all the experiments phrases are determined for this test set only. It is felt that the results achieved with this limited set will differ little from those of the full set. The use of a restricted set such as this is also a practical necessity since the great quantity of hand analysis required by these experiments precludes the use of the full document and query sets. Figure 1 indicates the results of a normal cosine retrieval process using the ten test queries. The following subsections discuss experimentation using various phrase determining techniques.

A) Statistical Phrases

The statistical phrase process uses a predetermined list of phrases.



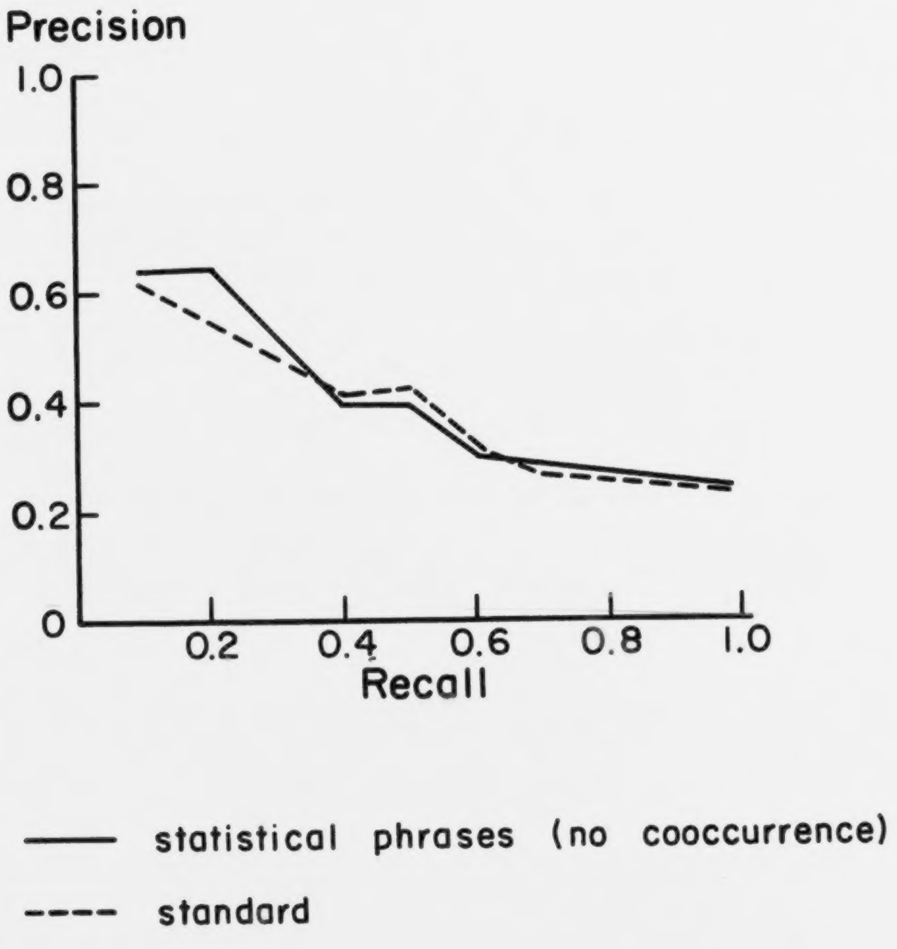
Standard Smart Results
(No Phrases)

Figure 1

The occurrence of the phrase elements in a document or query is considered an occurrence of that phrase regardless of the syntactic relation of the phrase components. A concept number is associated with a phrase and the appropriate concepts are appended to the document or query vectors. This method is clearly the simplest way to determine phrases since it requires no syntactic analysis of the text. However, statistical phrases have some serious drawbacks. Most obvious is the fact that they may recognize false phrases; that is, occurrences of the desired phrase elements but not in the proper syntactic relation. This problem can be minimized in small collections dealing with a narrow subject area by judicious selection of the statistical phrase list. In a corpus dealing with computer systems, for example, the occurrence of the words "real" and "time" can be viewed with relative certainty to be an occurrence of the phrase "real time". However as the collection grows and the subject area broadens, these decisions become less certain. Also the difficulty in creating the phrase list is increased as the corpus is enlarged. The phrase list can be determined by statistical means; however, weaknesses in this method can create problems. In the ADI collection for example, of the 409 statistical phrases in the test document set, only 153, roughly 37%, are syntactically correct. Figure 2 shows the results achieved using statistical phrases along with the standard no-phrase results. The results for statistical phrases are slightly higher in places, lower in others and show no significant overall improvement in retrieval quality.

B) Syntactic Phrases

As mentioned previously, almost two-thirds of the statistical phrases determined for the test set turn out to be syntactically incorrect. Removal



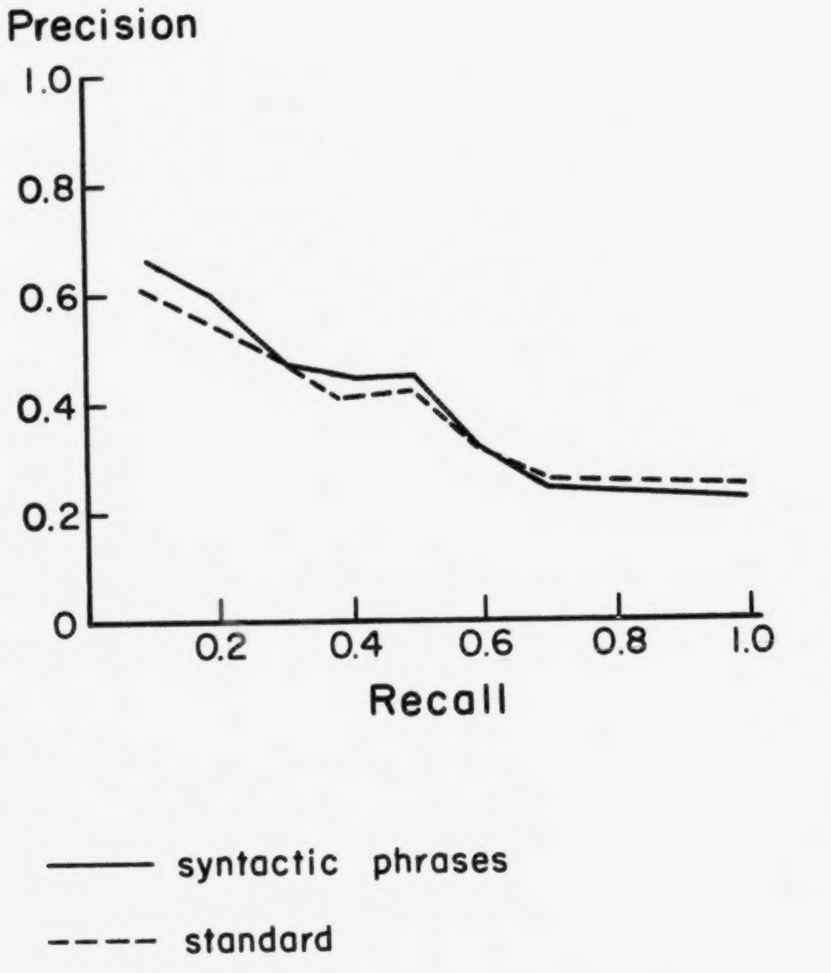
Experiment 2: Statistical Phrases

Figure 2

of the false phrases would allow the phrase component of the concept vector to represent more closely the true structure of the document or query. An automated process to perform this would first locate statistical phrases and then, using some syntactic analysis technique, weed out the erroneous ones. The syntactic analysis process required here is considerably simpler than general syntactic analysis since the process need only check the correctness of a statistical phrase rather than perform a complete syntactic parse. However, since the purpose of this study is to determine the value of syntactic phrases as a retrieval aid and not to test a syntactic analyzer, the analyses are done by hand. Removal of false phrases leaves 153 of the original 409 document phrases and 6 of the 12 query phrases. Results of this process are presented in Figure 3, and are again, disappointing. Statistical phrases show no significant improvement in retrieval performance.

C) Cooccurrence

The easiest way to handle phrases, and the way used in the previous experiments, is simply to assign each phrase a concept number and append the number onto the appropriate concept vector. After assignment, phrase concepts become indistinguishable from single word concepts, and the correlation coefficient operates normally. Unfortunately this gives rise to a number of serious problems. First, is the dilution effect caused by unmatched phrase concepts. The probability of a phrase match between a document and query is quite small due to the added structural requirements inherent in phrase matching. Furthermore since documents are typically much longer than queries, the document contains many phrases which cannot possibly match the query. As a consequence many phrase concepts are not matched. These unmatched concepts lower the correlation and partially if



Experiment 3 : Syntactic Phrases
(No Cooccurrence)
Figure 3

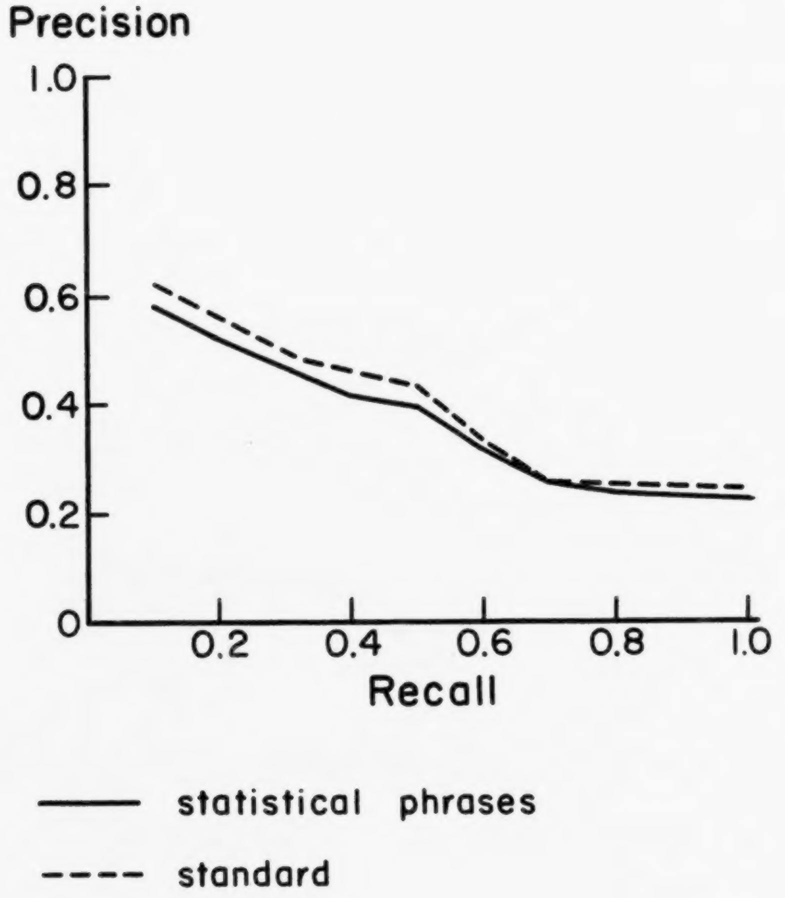
not completely offset any gain achieved by matched phrases. Thus the inclusion of too many phrases can dilute the vector with unusable information and inferior results may be produced.

A second problem deals with the value of a phrase as a nonrelevancy indicator. Individual word concepts are about equal as relevancy and nonrelevancy indicators. That is the cooccurrence of concept A in document D and query Q is as good a measure of D's relevance to Q as the lack of this cooccurrence is a measure of D's nonrelevance. As more structure is imposed on the comparison of documents and queries, cooccurrences become more significant but less frequent while non-cooccurring structures become less significant and more frequent. For example if documents are retrieved only if they match, word for word, the complete query, few if any documents would be returned. However any document which is retrieved by this scheme would almost certainly be relevant. On the other hand, the fact that some documents do not match the complete query is not a good indicator of their nonrelevance. The situation is similar for phrases. Thus treating phrase concepts simply as additional word concepts over-emphasizes their role as nonrelevancy indicators and while it may provide improved precision, it has disastrous effects on recall.

The problems presented above make it necessary to treat phrase and word concepts differently. In particular the role of phrases as a relevancy indicator must be weighted much more heavily than their role as a nonrelevancy indicator. The method designed to accomplish this is called cooccurrence matching and considers phrases only when they cooccur between a document and a query. Its operation may be seen from the following example. Let D and Q be the word concept vectors for a particular document and query, and PD and PQ, their associated phrase concept vectors. If phrase concepts are treated as word concepts, the correlation is calculated between D + PD and Q + PQ. The cooccurrence method on the other hand first calculates $C = PQ \cap PD$. That is, C is the set of phrase concepts common to both the query and document. Correlation is then calculated between D + C and Q + C. In this way it is guaranteed that phrase concepts cannot lower the correlation, and in the worst case where C is empty, the correlation is unaffected by the phrases. This process avoids the two previously discussed pitfalls associated with phrase use. First, by ignoring all unmatched phrase concepts, the vectors cannot become diluted with useless and possibly detrimental information. Secondly, phrases are used only as a relevancy indicator while their far weaker role of nonrelevancy indicator is not considered. The experiments performed in the remainder of this study all employ the cooccurrence principle for handling phrase concepts. The next two experiments are repeats of the previous two with the addition of the use of the cooccurrence phrase matching technique. The results are shown in Figures 4 and 5 and once again show no improvement over the no phrase method. A more complete analysis of these results is presented below.

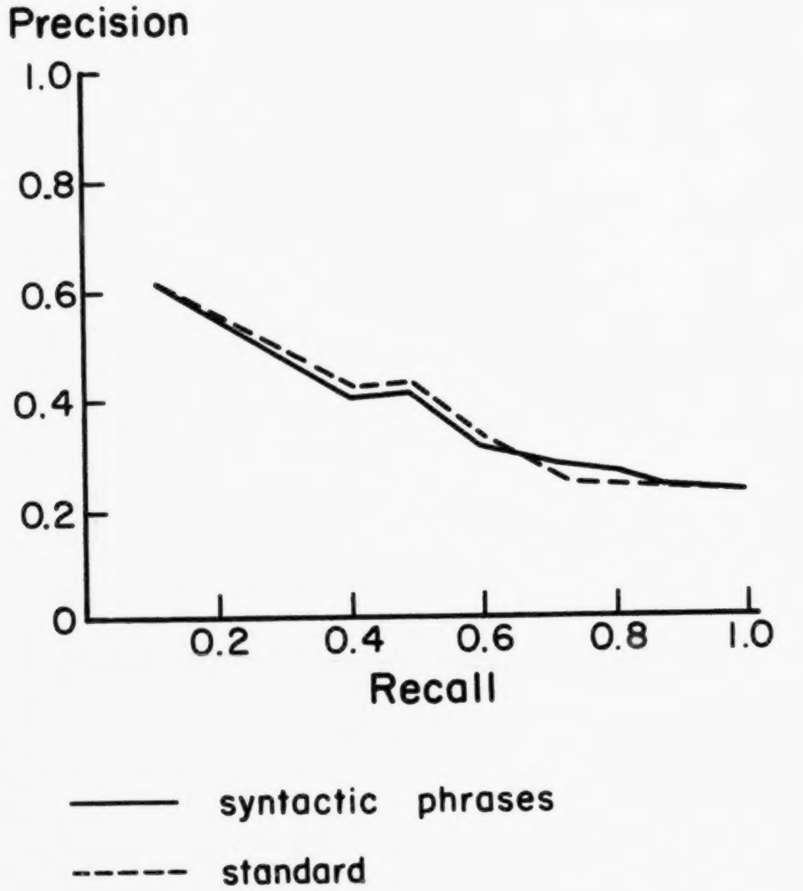
D) Elimination of the Phrase List

All methods discussed so far for using phrases in retrieval have required a phrase list. As previously mentioned the creation of these lists, whether by hand or by statistical processes, raises certain inherent problems. In general, it is far more desirable to be able to determine phrases without the need of such a list. One possible solution is to perform a syntactic analysis of the text, and determine all the phrases. The set of phrases thus generated is then normalized to associate all



Experiment 4: Statistical Phrases
(With Cooccurrence)

Figure 4



Experiment 5: Syntactic Phrases
(With Cooccurrence)
Figure 5

syntactically different but semantically identical phrases. This is accomplished, for example, by transformational kernelization of the phrases or by the use of a criterion tree matching scheme. Each phrase in the reduced set is then assigned a concept number, and retrieval proceeds as in the previous cases. However the syntactic analysis and normalization processes are prohibitively complex and produce a very large number of phrases. For these reasons an alternate method is used.

One of the easiest ways of accomplishing some degree of phrase processing without a phrase list is by means of the implicit phrase method. The philosophy behind this technique is that the cooccurrence in the document and query of several different concepts should be considered a better relevancy indicator than the cooccurrence of a single concept which has multiple occurrences and hence a higher weight. Consider the sample query and document vectors in Figure 6. The cosine correlation assigns the same correlation value to both. The second document however would seem to be more relevant to the query. The use of implicit phrases allows this fact to be reflected in the final correlation value. The basis of this process is a modified correlation coefficient formula:

$$C_{dq} = \frac{\sum_{i=1}^{N} d_i q_i + K(m-1)}{\left\{\left[\sum_{i=1}^{N} d_i^2 + K(m-1)\right]\left[\sum_{i=1}^{N} q_i^2 + K(m-1)\right]\right\}^{\frac{1}{2}}}$$

where m is the number of different concepts which cooccur in the document and query, and K is a constant. In the general case K = 144*P where P is an experimental parameter. In this way each pair of cooccurring concepts in the document and query is treated as a phrase and the correlation is is treated accordingly. In Figure 6 for example, the implicit phrases

QUERY: INFORMATION RETRIEVAL

DOC-1 INFORMATION ABOUT INFORMATION
DOC-2 INFORMATION RETRIEVAL AND SYSTEMS ANALYSIS

VECTORS:

	INFORMATION	RETRIEVAL	SYSTEMS	ANALYSIS	CORRELATION WITH QUERY
QUERY	12	12			
DOC-1	24				0.786
DOC-2	12	12	12	12	0.786

Sample Document and Query Vectors

Figure 6

correlation between document 1 and the query remains unchanged while the correlation of document 2 is raised to 0.774 thus reflecting its apparent greater relevancy. Figure 7 shows the results of retrieval using the ADI collection and the implicit phrase process with various values for P. It indicates that some improvement is achieved over the no-phrase process. However, one of the main drawbacks of the process is that it fails to fulfill one of the primary objectives for phrase use. That is it cannot discriminate between documents with similar concepts but different structural relationships among these concepts. For this reason a more syntactically oriented approach to phrase processing must be used.

The syntactic process used is relational content analysis. This process determines syntactic relations between pairs of text words. The details of relational content analysis are discussed by Weiss [9]. Concepts which are determined to be related by the content analyzer are encoded into a special phrase concept number, XXXXYYYYZZ, where XXXX represents the concept number of the first word, YYYY the second, and ZZ is the relation between them. The order of the two concepts is significant for all relations except parallel in which the smaller concept number appears first. The encoded relational phrases are treated as concept numbers and assembled into a phrase concept vector. The phrase vector must be kept separate from the word vector to permit the use of the cooccurrence phrase matching process. The retrieval results for this technique with the ADI test set appear in Figure 8.

Using this type of process for phrase determination has a number of advantages. First, it alleviates the need for an a priori phrase list. Also, being a relatively simple process, it has significantly more practical value than some of the more complex systems. Clearly a great deal of

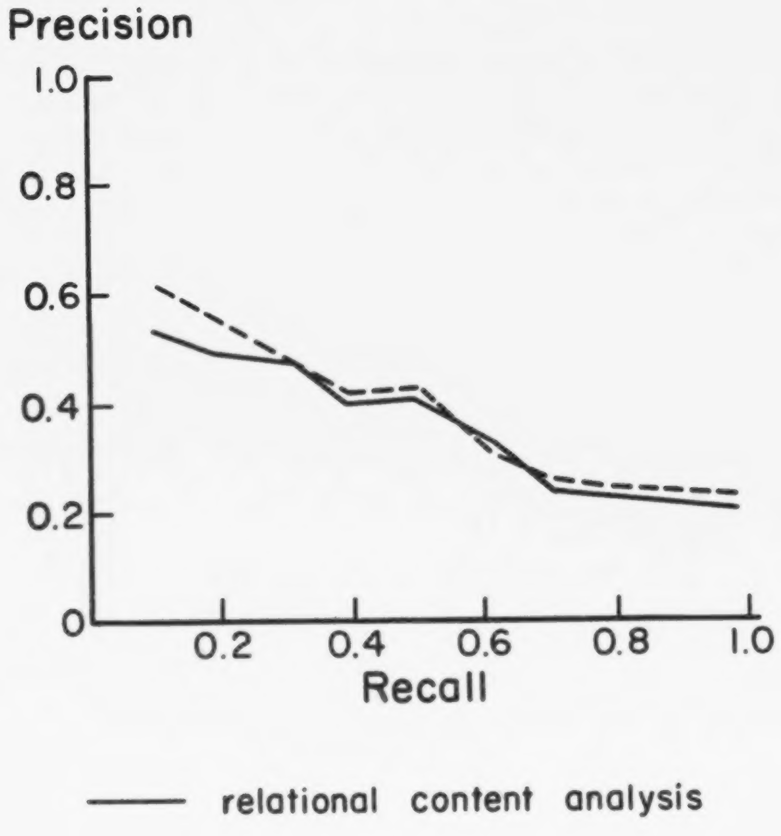
RECALL	IMPLICIT PHRASE TRIAL 1	2	3	4
0.1	.6124	.6333+	.6310+	.6278+
0.2	.5524	.6333+	.6310+	.6278+
0.3	.4862	.5643+	.5643+	.5577+
0.4	.4286	.4392+	.4356+	.4291+
0.5	.4335	.4309-	.4273-	.4255-
0.6	.3351	.3441+	.3341-	.3341-
0.7	.2608	.2651+	.2565-	.2538-
0.8	.2569	.2682+	.2597+	.2570+
0.9	.2493	.2590+	.2549+	.2427-
1.0	.2493	.2590+	.2549+	.2427-

1. = standard, no phrases
2. = implicit, p=1.0
3. = implicit, p=1.5
4. = implicit, p=2.0

+ indicates better than trial 1
- indicates worse than trial 1

ADI with Implicit Phrases

Figure 7



Experiment 7: Relational Content Analysis

Figure 8

syntactic information is lost since only word pairs are considered however, cooccurrences in documents and queries of syntactic structures more complex than word pairs is exceedingly rare. Thus despite its simplicity, relational content analysis does perform the particular aspect of syntactic analysis most relevant to information retrieval. Besides the advantages there are also some disadvantages inherent in this type of system. Most serious is its inability to associate semantically similar phrases. A system that uses a phrase list can recognize equivalent phrases whose constituent concepts are not equivalent. For example, the phrases "memory holding" and "data processing" are both assigned the same phrase concept by the SMART phrase list for the ADI collection, while each of the four words falls into a different concept class. The recognition of such equivalent phrases is impossible for systems which do not employ such a list of extensive semantic normalization. It may therefore be expected that retrieval results achieved by the relational concept analyzer will be inferior to those achieved in previous experiments. However, retrieval without the requirement of a phrase list seems to be a more reasonable approach to the problem. This is especially true in the case of large document collections where manual creation of a phrase list is impossible and statistical creation in unreliable.

E) Analysis of ADI Results

The results of the seven retrieval experiments are summarized in Figure 9. The plus or minus to the right of each figure indicates whether if is above (+) or below(-) the standard no-phrase value achieved for that recall level, (experiment 1). The results clearly show that there is no great gain achieved by the use of phrases and in some cases their

R	1	2	3	4	5	6	7
0.1	.6124	.6258+	.6500+	.5876-	.6124	.6333+	.5458-
0.2	.5524	.6258+	.6000+	.5276-	.5524	.6333+	.4858-
0.3	.4862	.4957+	.4798-	.4639-	.4826-	.5643+	.4862
0.4	.4287	.4078-	.4423+	.4223-	.4244-	.4392+	.4280-
0.5	.4335	.4059-	.4470+	.4096-	.4327-	.4309-	.4327-
0.6	.3351	.3234-	.3312-	.3208-	.3338+	.3441+	.3376+
0.7	.2608	.2742+	.2547-	.2608	.2617+	.2651+	.2586-
0.8	.2569	.2782+	.2426-	.2555-	.2571+	.2682+	.2506-
0.9	.2493	.2675+	.2346-	.2433-	.2492-	.2590+	.2435-
1.0	.2493	.2675+	.2346-	.2433-	.2492-	.2590+	.2435-

1. = standard
2. = statistical, no occurrence
3. = syntactic, no cooccurrence
4. = statistical, cooccurrence
5. = syntactic, cooccurrence
6. = implicit, p=1
7. = relational

Summary of Phrase Method Results

Figure 9

use appears to be actually detrimental. However, upon more careful analysis of these results, a number of unusual factors are found which make these results somewhat less discouraging than they initially appear.

Consider first the results obtained with the statistical and syntactic phrases. It is argued in section C that the use of cooccurrence improves the retrieval quality. The results seem to indicate that exactly the opposite is true for experiment 4 and that experiment 5 results exceed experiment 3 at only half of the recall points. Upon analysis of the retrieval output it is discovered that the reason for this apparent turnabout is the dilution of nonrelevant concept vectors due to unmatched concepts. For many of the queries analyzed, there is one or more documents, highly correlated to that query, but nonrelevant, and which has a relatively large number of phrases which are not matched in the query. Because of the dilution effect which occurs when cooccurrence is not used, the correlations for these documents are lowered, often to a level below that of one of the relevant documents. The rank of the relevant document is thus raised by default even though its own correlation is not altered. Consider for example the correlation of document 11 with query A4. With no phrases used, this nonrelevant document ranks sixth with a correlation of 0.248189. The document has 13 statistical phrases which do not match the query. When retrieval is performed using these phrases without cooccurrence, the coefficient is reduced to 0.15599 and the rank lowered to ninth place. This allows one of the relevant documents to move ahead producing an apparent improvement in retrieval quality. When cooccurrence is used there are no phrase matches, the coefficient remains 0.24818, and the relevant document is not allowed to move up. Considering the entire set of 33 documents relevant to the test queries, the ranks of 16 are improved by the use of statistical phrases with no cooccurrence. However, of these, only 7 actually move up in correlation coefficient. The remaining 9 lose in correlation but gain in rank due to the dilution and consequent lowering of nonrelevant documents. Ten of the 33 relevant documents lose in both rank and coefficient, mostly due to being diluted themselves, while 7 remained fixed in rank. Of these 7, 5 are reduced in coefficient but by an amount insufficient to drop the rank. Also most of the documents with a large number of phrases are not relevant to any test query. Thus the apparent superiority of the no-cooccurrence process (experiments 2 and 3) over the normal method (experiment 1) and the cooccurrence process (experiments 4 and 5) is almost entirely due to the lowering of the correlation coefficient of certain nonrelevant documents. This in turn is aided by the fact that most documents with a large number of phrases are not relevant to any query. The reduction in rank of these documents with respect to any query is thus guaranteed to cause, at worst, no harm and possibly produce a default raise in rank of a relevant document. This situation is clearly not typical. In general, every document must be considered as a potential relevant document. Lowering the rank for some set of documents for all queries would thus help retrieval in some cases, harm it in others. The results of experiments 2 and 3 reflect some positive effect caused by increasing the correlation in relevant documents. However, this effect is quite small. In general it can be concluded that since the conditions which led to the results of experiments 2 and 3 cannot be considered typical of document and query collections, the apparent improvement in retrieval quality achieved with no-cooccurrence must therefore be held suspect.

Attention is next focused on experiments 4 and 5 which use statistical and syntactic phrases with the cooccurrence technique. When compared with experiment 1, the results seem to indicate that the cooccurrence processes are harmful to retrieval quality. However, this result is misleading as a result of a peculiar situation. This can be understood by considering the results of experiment 4. Of the 33 relevant documents, this phrase process improves both the rank and correlation for 9; 5 are reduced in rank; while the remaining 19 are unchanged. Overall this seems to be an improvement, but the tabulated results in Figure 9 do not bear this out. The reason for this lack of improvement lies almost entirely with query B5. It has only one relevant document and the phrase process lowers its rank from second to fifth thus lowering its precision for all recall levels from 0.5 to 0.2. This is a considerable decrease in precision, and since the values are averaged over only ten queries, the effect on the average is substantial. If precision values are taken for the nine other queries only, the values for the phrase processes exceed those for the no-phrase experiment for nearly all recall levels. Thus except for a rather unusual query, these phrase processes using cooccurrence provide some degree of improved retrieval results. The main drawback of such a process is the need for an a priori phrase list. And it is for this reason that the major emphasis in this study is on phrase methods which do not require predetermined lists.

The tabulations in Figure 9 indicate that results achieved by using the no-phrase-list method based on relational content analysis (experiment 7) are inferior to both the phrase list and no-phrase results. This is in part due to the method's inability to associate phrases with different constituent concepts. The inferior results can also be blamed on the very small number of cooccurrences. Of the more than 800 relations entered, only 28 cooccurrences between documents and queries are found. This very low number can be blamed, at least in part, on the queries. They are all quite short and contain very few phrases. The queries also tend to be quite general. Since retrieval is performed by concept matching and not by hierarchical expansion, general queries do not always produce the desired results. Of the 28 cooccurrences, only 5 occur between a query and one of its relevant documents. In the ten test queries, three have no cooccurrences at all, and their results are clearly not altered from the no-phrase case. Four queries have cooccurrences in nonrelevant documents only and these results are obviously lowered. The three remaining queries have cooccurrences in relevant documents; however an improvement is realized in only one. Of the other two, one shows an improvement in correlation coefficient, but insufficient for a rank change, and the other has cooccurrences in nonrelevant documents which overshadow any improvement. These results might appear to cast some doubt on the value of this method. However this evidence is inconclusive and thus any decision is premature.

From the previous experiments it appears that the various phrase and structure methods can provide some degree of improvement in retrieval quality. But this improvement may be insufficient to warrant the additional work needed to use them. This deficiency, however, cannot be blamed entirely on weaknesses in the methods used. In the introduction to this study one of the primary uses of phrases in information retrieval is stated to be the separation of highly correlated, but not semantically identical, documents. A document collection must therefore contain such close documents in order for phrases to demonstrate any significant retrieval improvement. To determine if the ADI collection provides a fair testbed for phrase use, a document-document correlation is preformed. The results indicate an average document-document correlation of 0.1 and a maximum of 0.8. This indicates that the ADI document space is in general quite sparse; but it may still contain some dense clumps of documents. To test for this, a third statistic is calculated; the average maximum document-document correlation (AMC). This is the correlation between a given document and its nearest neighbor averaged over all document-document pairs. In the ADI collection the AMC is less than 0.4 thus indicating the general absence of dense document clumps. Thus the documents in the ADI collection are seen to be quite spread out in the document space; and the extra dimension of refinement added to the documents and queries by the use of syntax is superfluous. Therefore to test more conclusively the usefulness of phrases in information retrieval, a more dense collection must be tried. Experiments with various other collections comprise the remainder of this study.

3. The Cranfield Collection

The Cranfield-424 Collection is a set of 424 documents in the field of aerodynamics. Because of its single specialized theme it might conceivably provide a denser collection on which to perform phrase experiments. Unfortunately this is not the case. Results of a document-document correlation are effectively the same as those for the ADI. The average document-document correlation is less than 0.1 and the AMC is about 0.4. It may therefore be expected that the Cranfield and ADI share the same undesirable characteristics concerning phrase use. For this reason the Cranfield collection is not used in this study.

4. The TIME Subset Collection

A) Construction

Because the existing collections do not exhibit the desired characteristics for conclusive testing of phrase techniques, a new collection is constructed. The process for creating such a collection is as follows. From an existing set of documents and queries, a subset of closely related queries is chosen. The set of documents relevant to any query in the subset is taken as the new document collection. The fact that these documents are all relevant to closely related queries guarantees that the documents themselves are also highly correlated. The collection chosen for this study is a set of articles from the "World" section of "TIME Magazine" (1963) with an associated set of current events queries. The largest number of related queries is six which deal with the Viet Nam war and particularly with the religious and political strife leading up to the overthrow of the Diem government. A total of 27 documents are relevant to these queries and this forms the TIME subset collection. The relatively small size of this document set detracts somewhat from the significance of the results of experiments using it, but not as much as might be expected. This is true for several reasons. First, the subset can be thought of as a single cluster in a large clustered document set. Since the subset contains all of the Viet Nam articles, its cluster centroid would clearly correlate highly with any Viet Nam related query. The real retrieval problem than becomes picking the desired articles from within the cluster. And second, the purpose of this set is to test the usefulness of phrases in information retireval, and phrases are micro rather than macro information retrieval aids. That is, the primary use for phrases is in determining fine differences in closely related documents, and not in producing tremendous rank increases for low ranking documents. Thus this type of collection is sufficient for testing phrase processes.

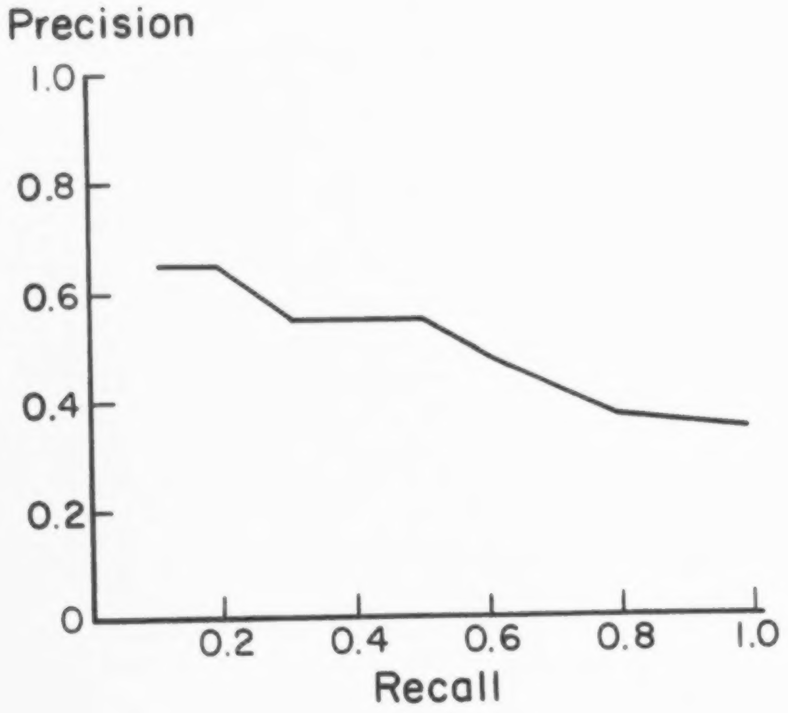
The TIME articles are written in a very conversational and chatty style as opposed to the technical style fo the ADI and Cranfield collections For example, a document dealing with the Vietnamese coup begins:

> Coping with Capricorn in business, count the costs before you act. The moon now in Capricorn suggests keeping practical values in mind. Tomorrow is rather too energetic for comfort, but that may be because everybody is on the move. (A late August horoscope.) Syndicated horoscopes, many of them from abroad are a popular feature in many South Vietnamese newspapers, but last week the government banned them, presumably on a theory that some star-minded dissident might be moved to try a coup on an astrologically auspicious day.
> ["TIME", 9/6/63, page 19]

The article then presents its true purpose, that of describing the increasing United States dissatisfaction with the present South Vietnamese government and the possibility of an American-encouraged coup. The article goes on by describing the martial law measures being taken by the Vietnamese government to prevent a coup, and then gives a brief biography of several generals who might stage the coup. Thus the crux of the article is to describe the tenuous political situation in Viet Nam, not to discuss astrology. The paragraph quoted above thus serves merely as a light introduction.

Construction of document vectors from the full text of articles such as this could very well result in a tremendous amount of spurious information in the vector. For this reason, and because of the document length, it is necessary to form abstracts. The abstracts used are about one hundred words in length and present the main ideas of the article using much the same vocabulary and constructions as in the original text. The abstracts thus capture the gist of the article in both content and style while eliminating most of the unrelated chaff. Using these abstracts, a vocabulary is constructed and document vectors are formed using standard SMART dictionary construction and vector creation programs. The dictionary assigns a single concept number to all words with a common stem. Figure 10 presents the results of a normal SMART search with the TIME subset collection. The results are consistent with retrieval results using other collections. There thus seems to be nothing particularly unusual about this document and query set which might tend to diminish the significance of any experimental results.

Three sets of phrase experiments are performed using the TIME subset collections. The first two are the implicit and relational as presented earlier. As before, various parameters are used to weight the importance of a phrase match in the correlation calculation. A third phrase process called half relational is also used. This is a weaker form of relational phrase matching (heretofore referred to as full relational for clarity). In full relational, a phrase match occurs only when the document and query have the same concept pair and the concepts are joined by the same relation. In Figure 11 below, the query phrase QP matches only document phrase DP1. In half relational matching, a match occurs when the document and query share a concept which occurs in the same relational context in both vectors. For example in Figure 11, the query QP matches document phrases DP1, 2, and 3 but not 4. While the query concept matches in DP4, the relational context does not. That is, in QP concept 5 is a modifier while in DP4 it is modified. Thus as the name implies, half relational matches require only one of the two related concepts to match. This is clearly a weaker matching requirement and is expected to produce more matches than full relational. This



Standard Results, Time Subset

Figure 10

could be of value in cases where cooccurrences of whole phrases are rare; but it may also give many improper matches.

QP < 5, 7,MOD>

DP1 < 5, 7,MOD>
DP2 < 5, 9,MOD>
DP3 <13, 7,MOD>
DP4 < 3, 5,MOD>

Sample Query and Document Relations Phrases

Figure 11

The results for these experiments are shown in Figure 12 A, B, and C. Figure 13 gives the tabulated results for each method using the weighting parameter which provides the best results. While these represent the best values, the results achieved for other parameter values are only very slightly lower. As before the figure shows whether the results of the phrase experiment are above (+) or below (-) those achieved when no phrases are used. These results reveal that implicit phrase matching is harmful to retrieval quality and gets worse as the weighting parameter is increased. Half relational shows some slight improvement for low recall values while full relational is generally worse. However in these latter two methods, all differences are very small and effectively insignificant.

B) Analysis of Results

The most surprising result of this set of experiments is the harmful effect caused by implicit phrases. This is inconsistent with the results obtained with the ADI collection. This apparent turnabout can be explained by recalling the original purpose for using implicit phrases. This is to separate those documents whose correlation is based on a cooccurrence of

RECALL	STANDARD	TIME IMPLICIT PHRASES			
		P = 0.5	P = 1.0	P = 1.5	P = 2.0
0.1	.6426	.6333-	.5639-	.5635-	.5635-
0.2	.6426	.6333-	.5639-	.5635-	.5635-
0.3	.5537	.5778+	.5639+	.5635+	.5635+
0.4	.5500	.5361-	.5125-	.5135-	.5135-
0.5	.5500	.5361-	.5125-	.5135-	.5135-
0.6	.4781	.4604-	.4447-	.4429-	.4429-
0.7	.4217	.4215-	.4256+	.4183-	.4183-
0.8	.3745	.3652-	.3564-	.3579-	.3579-
0.9	.3702	.3577-	.3555-	.3496-	.3496-
1.0	.3669	.3577-	.3555-	.3496-	.3496-

Summary of TIME Implicit Phrase Experiments

Figure 12A

RECALL	STANDARD	TIME FULL RELATIONAL PHRASES			
		P = 0.5	P = 1.0	P = 1.5	P = 2.0
0.1	.6426	.6389-	.6359-	.6333-	.6333-
0.2	.6426	.6389-	.6359-	.6333-	.6333-
0.3	.5537	.5500-	.5803+	.5778+	.5778+
0.4	.5500	.5417-	.5215-	.5190-	.5190-
0.5	.5500	.5417-	.5215-	.5190-	.5190-
0.6	.4781	.4614-	.4520	.4578-	.4634-
0.7	.4217	.4079-	.4041-	.4099-	.4154-
0.8	.3745	.3632-	.3602-	.3577-	.3577-
0.9	.3702	.3632-	.3602-	.3577-	.3577-
1.0	.3669	.3632-	.3602-	.3577-	.3577-

Summary of TIME Full Relational Phrase Experiments

Figure 12B

RECALL	STANDARD	TIME HALF RELATIONAL PHRASES			
		P = 0.5	P = 1.0	P = 1.5	P = 2.0
0.1	.6426	.6274-	.6274-	.6274-	.6663+
0.2	.6426	.6274-	.6274-	.5857-	.6107-
0.3	.5537	.5218-	.5163-	.5718+	.6107+
0.4	.5500	.5218-	.5112-	.5649+	.5788+
0.5	.5500	.5218-	.5112-	.5649+	.5788+
0.6	.4781	.4448	.4362-	.4468-	.4468-
0.7	.4217	.4111-	.4062-	.4111-	.4111-
0.8	.3745	.3395-	.3350-	.3259-	.3198-
0.9	.3702	.3395-	.3350-	.3259-	.3198-
1.0	.3669	.3372-	.3327-	.3236-	.3175-

Summary of TIME Half Relational Phrase Experiments

Figure 12C

RECALL	STANDARD	IMPLICIT P = 0.5	FULL P = 0.5	HALF P = 2.0
0.1	.6426	.6333-	.6389-	.6333+
0.2	.6426	.6333-	.6389-	.6107-
0.3	.5537	.5778+	.5500-	.6107+
0.4	.5500	.5361-	.5417-	.5788+
0.5	.5500	.5361-	.5417-	.5788+
0.6	.4781	.4604-	.4614-	.4468-
0.7	.4217	.4215-	.4079-	.4111-
0.8	.3745	.4652-	.3632-	.3198-
0.9	.3702	.3577-	.3632-	.3198-
1.0	.3669	.3577-	.3632-	.3175-

Summary of TIME Processes

Best Results Used for Each

Figure 13

several concepts in the document and query from those documents whose correlation results from one or two highly weighted concepts. In the ADI collection, there are many concepts in the documents with weights of twenty-four or more so that there is a real need for such a separation technique. As a result, implicit phrases provide improved retrieval for the ADI. In the TIME collection occurrences of highly weighted concepts are much rarer than in the ADI. Consequently the reason for using implicit phrases does not exist. Employing the phrase technique thus does not accomplish the purpose for which it is designed and hence no improvement is realized. Thus it appears that implicit phrases may be a useful technique but only when used with collections which meet certain requirements as to the presence of highly weighted concepts.

The results achieved using both half and full relational content analysis are discouraging. They may be the result of weakness in the phrase process or, as in the case of the ADI collection, they may be caused by the collection itself. Figure 14 shows for each method how many phrases are matched with relevant and nonrelevant documents. In both cases only about one-third of the phrase matches are between a query and one of the relevant documents. This seems to indicate that the weakness may lie in the phrase matching method, however this is only partially true. The reason for the poor results for the half relational is simply that the matching criteria are too weak. Too many false and incorrect phrases are matched and the lower retrieval quality results. It therefore seems the half relational method is worthless although some further testing is necessary to finalize the decision. The reason for the poor results with the full relational method is not so clearly the fault of the matching scheme. Of the 82 phrase matches between documents and queries, 65, or roughly 80%, are matches of the phrases "South Viet" or "Viet Nam". Since the entire collection deals with South Viet Nam, these phrases occur almost uniformly throughout the document set. And since each query has an average of three times as many nonrelevant as relevant documents, the results in Figure 14 are to be expected. If this document collection were considered as one cluster of a larger collection, the phrase South Viet Nam would be useful in gaining access to the cluster. However, within the cluster it is a poor discriminator and thus cannot help retrieval. If South Viet Nam is removed from the set of phrase matches, more than two-thirds of the remaining phrase matches occur between a query and a relevant document and retrieval would clearly be improved. However the small number of relations that remain seem to indicate the same collection sparseness as is found in the ADI and Cranfield collections.

	Number of Phrase Matches				
	With Rel Documents		With Nonrel Documents		Total
Half Relational	89	32%	186	67%	277
Full Relational	28	34%	54	65%	82

Phrase Matches (TIME)

Figure 14

A document-document correlation on the TIME subset collection reveals that the average correlation is 0.2. This is twice as high as the ADI or Cranfield and is to be expected since the TIME collection is designed specifically for high density. However, the average maximum correlation (AMC) which is a more important measure is 0.41, roughly the same as for previous collections. This indicates that the increased density in the collection is achieved by the omission of low correlating documents, and not by the occurrence of highly correlated document pairs. And this collection is seen as no better for phrase experimentation than the ADI. Thus is appears that even though this collection is constructed specifically for phrase use, it does not satisfy some of the theoretical prerequisites. The natural question at this point is exactly in what type of collection are phrases useful. This question is treated in the next section.

Beside collection density, there is another factor affecting the usefulness of phrases. This is the type of relations occurring between text elements. There are basically two types of semantic relations by which phrase words may be associated: reversible and nonreversible. A reversible relation is one in which the ordering of the constituent words has no effect on the meaning. For example the words "information" and "retrieval", occurring in almost any structure means "information retrieval", and hence the words are related by a reversible relation. A nonreversible relation is one in which the phrase structure is significant. The relation between "U. S." and "Russia" in the sentence below is an example of a nonreversible relation.

The U. S. influences Russia.

There is also a third type of relation, which is usually a specialized subset of nonreversible, called trivial nonreversible. These are phrases whose meaning depends on the structure and are technically nonreversible.

However, with these special phrases, all but one of the potential meanings do not occur in practice, and the relation assumes reversible characteristics. For example, consider the sentence:

The U. S. invades Cambodia.

Since it is possible for the U. S. to invade Cambodia and vice versa, the relation between U. S. and Cambodia is clearly nonreversible. However, since in fact Cambodia has not and probably will never invade the United States, the relation is actually trivial nonreversible and hence its structure becomes unimportant. As mentioned earlier, one of the primary objectives of the use of structured phrases is in matching phrases whose meaning is a function of both its content and its structure, that is, phrases with nonreversible relations. If such phrases do not occur in the analyzed text, structured phrase use can clearly provide little or no help in retrieval. This is the case in the TIME collection. Of the phrases isolated, a vast majority are reversible or trivial nonreversible. Thus the lack of nonreversible relations is another reason for the failure of the content analysis scheme to achieve improved results.

5. A Third Collection

In the previous sections it is shown that the ADI and TIME collections do not require the use of phrases because they do not demonstrate the characteristics which provide the theoretical basis of phrase use. They are neither dense enough nor do they contain large numbers of nonreversible relations. And hence no significant advantage is gained through the use of phrases. Analysis of other natural collections such as the Cranfield reveals the same situation. The natural question at this point is this: what is a collection like which has the desired characteristics? To attempt to answer this a purely artificial collection is constructed. The collection consists of twenty documents and fourteen queries, each in the form of a short sentence. The subject matter deals with the relation between birds and worms and is inspired by an example by Simmons [8]. This highly specific subject guarantees a highly dense document space. In addition, the documents are specifically written to include nonreversible relations. For example, in

Birds eat worms.
Worms eat grass.

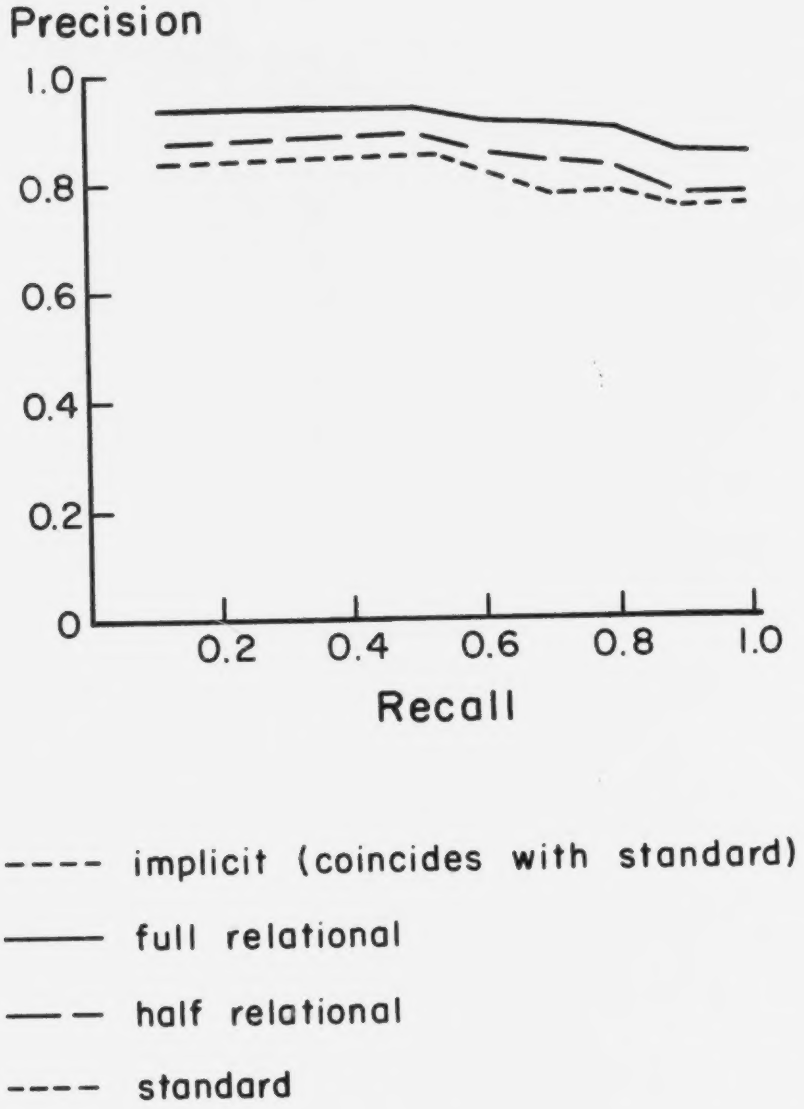
The words "worms" and "grass" are clearly nonreversibly related. This collection might thus be considered an ideal testbed for phrase experimentation.

Results are tabulated in Figure 15 and shown graphically in Figure 16. Because of the extreme closeness of the various results, only the best of each set is shown. Also the results of implicit phrases are not shown on the graph in Figure 16 since they coincide with the no phrase results. The lack of improvement here is caused, as in the TIME collection, by the lack of highly weighted concepts in the document and query vectors. Thus the problem which implicit phrases are designed to solve simply does not exist. The results for half relational phrases show a slight improvement at all recall levels. More important, however, are the results in Figure 17. This indicates that only about a third of the half relational phrase matches are between a query and one of its related documents. This seems to finalize the conjecture stated earlier that half relational matching is too weak a criterion and results in too many improper phrase matches.

RECALL	STANDARD	IMPLICIT	FULL	HALF
0.1	.8440	.8440	.9286+	.8810-
0.2	.8440	.8440	.9286+	.8810-
0.3	.8440	.8440	.9286+	.9810-
0.4	.8440	.8440	.9286+	.8810-
0.5	.8440	.8440	.9286+	.8810-
0.6	.8083	.8383	.9000+	.8524+
0.7	.7798	.7798	.9000+	.8524+
0.8	.7798	.7798	.8929+	.8333+
0.9	.7548	.7548	.8393+	.7554+
1.0	.7548	.7548	.8393+	.7554+

Summary of B&W Phrase Processes

Figure 15



B & W Phrase Results

Figure 16

It thus appears to be an unsuitable phrase process. As Figure 17 indicates, quite the opposite is true for full relational phrases. More than two-thirds of the full relational phrase matches are with relevant documents. This fact is also reflected in the improved precision at all recall levels achieved by any full relational matching. These results can be treated both optimistically and pessimistically. On the one hand, they show conclusively that structural phrases can be of value in information retrieval. On the other hand, this improvement in retrieval results is not achieved in "natural" collections such as the ADI, but rather only for one which is highly artificial and contrived. It is not clear at this point whether any natural collection can meet all of the requirements for advantageous phrase use.

6. Conclusion

The general conclusions that can be drawn from these experiments are that a number of different types of phrase processes are useful in information retrieval provided certain characteristics exist in the document set. This is especially true in the case of structural phrases where it appears that effective phrase use depends more on the collection than on the specifcc phrase process.

The implicit phrase process is designed to boost correlations based on the cooccurrence of many concepts in the document and query as opposed to those correlations which are the result of a very few matches of highly weighted concepts. Results indicate that it performs the job quite well. However, if the collection has relatively few high weights, the need for implicit phrases no longer exists. Using implicit phrases with such collections is thus a wasted effort and may even lead to downgraded retrieval quality.

	NUMBER OF PHRASE MATCHES				
	WITH REL DOCUMENTS		WITH NONREL DOCUMENTS		TOTAL
HALF RELATIONAL	62	38%	102	62%	164
FULL RELATIONAL	36	69%	16	31%	52

Phrase Matches (B & W)

Figure 17

For structured phrases to be of value in information retrieval, a number of conditions must be met. First the collection must be sufficiently dense, or at least have some dense clumps of documents. Second, the document must contain nonreversible relations. Along the same line, the documents in any particular clump must be sufficiently different semantically so that conceivably some but not all could be relevant to a given query. In other words, there must be a potential need to discriminate between closely related documents. This restriction is necessary for the following reason. It is conceivable that a particular clump of documents could be so closely related that either all or none are related to any query. While this clump satisfies the density requirement and may have nonreversible relations as well, it does not require the use of phrases. There is no need to distinguish among members of the clump and thus phrases cannot help. Finally, it is necessary that the queries contain nonreversible relations. If such relations are not requested in the query, as is true in the ADI collection, no advantage is gained by using them in the documents. Testing this condition is easy when dealing with experimental documents and queries, but clearly impossible in real applications. However, it is possible to predict the general form for expected queries and thereby determine if they meet the phrase requirement. As a general guideline, queries are more applicable to phrase use if they are of the question-answering variety rather than pure document retireval.

The final conclusion that is reached from this study is that, contrary to intuition, phrases do not seem to exert a large effect on a user choice of relevant documents. Future work must be done on determining the factors that go into a user's relevancy decisions. With more insight into this area, the role of structure in information retrieval will become much more clearly defined.

References

[1] Curtice, R. M., and Jones, P. E., An Operational Interactive Retrieval System, Arthur D. Little, Inc., 1969.

[2] Douglas, E., Mandersloot, W., and Spicer, N., Thesaurus Control — the Selection, Grouping, and Cross-referencing on Terms for Inclusion in a Coordinate Index Word List., Journal of the American Society for Information Science, January-February, 1970.

[3] Hutchins, W. J., Automatic Document Selection Without Indexing, Journal of Documentation, Vol. 23, No. 4, December 1967.

[4] IBM Systems/360 Document Processing System, Applications Description, IBM, 1967.

[5] Lesk, M. E., A Proposal for English Text Analysis, Bell Telephone Laboratories, 1969.

[6] Salton, G., Automatic Information Organization and Retrieval, McGraw-Hill, New York, 1968.

[7] Salton, G., Automatic Text Analysis, Science, Vol. 168, April 17, 1970.

[8] Simmons, R. F., Synthex. In Orr, W. D., (Ed.), Conversational Computers, John Wiley and Sons, Inc., New York, 1968.

[9] Weiss, S. F., A Template Approach to Natural Language Analysis for Information Retrieval, Ph.D. Thesis, Cornell University, Ithaca, New York, 1970.

II. The "Generality" Effect and the Retrieval Evaluation for Large Collections

G. Salton

Abstract

The retrieval effectiveness of large document collections is normally assessed by using small subsections of the file for test purposes, and extrapolating the data upward to represent the results for the full collection. The accuracy of such an extrapolation unhappily depends on the "generality" of the respective collections.

In the present study the role of the generality effect in retrieval system evaluation is assessed, and evaluation results are given for the comparison of several document collections of distinct size and generality in the areas of documentation and aerodynamics.

1. Introduction

Over the past few years a great many studies have been undertaken in an attempt to assess the retrieval effectiveness of a variety of automatic analysis and search procedures. Under normal circumstances, a single test collection is used which is subjected to a variety of processing methods; paired comparisons are then made between two or more procedures for this collection in order to determine which methods are most effective in a retrieval environment. [1,2,3]

Occasionally, however, it is necessary to use several different document collections in a test situation and to compare the results for distinct collections (rather than for distinct processing methods). Such is the case notably when a variable is tested for which a single collection is not normally usable (for example, the language in which the documents are written [4]), or when an attempt is made to extrapolate from a small test collection to a large operational one. [5] In such situations, special precautions are needed to insure that the evaluation measures actually reflect the performance differences between the respective collections.

Consider as an example, two distinct document collections. Performance differences might then emerge as a result of the following collection characteristics:

a) differences in subject matter;
b) differences in the scope of the collections;
c) differences in the document types available for processing;
d) differences in query types;
e) differences in the collection size;
and f) differences in the relevance judgments of queries with respect to documents.

In the present study, the first four variables are not under investigation in the sense that comparisons are made only for collections of document abstracts of similar scope within a specific subject area, using standard user requests of the type often submitted to an information center. The other two variables, namely collection size and type of relevance assessments are of special interest, since both of them affect the evaluation results obtained for large operational systems. These variables to a large extent determine the generality of the collection, that is, the average number of relevant items per query, and generality in turn affects the evaluation parameters.

In the remainder of this study, two different generality problems are examined by using on the one hand collections of different size for which the relevance judgments agree, and, on the other hand, collections of identical size with different relevance properties. The variations obtained in the evaluation results are examined, and an attempt is made to interpret the respective performance differences.

2. Basic System Parameters

The evaluation parameters used to assess the retrieval performance of a given set of user queries with respect to a document collection are normally based on a two by two contingency table which distinguishes between the documents retrieved in answer to a given query and those not retrieved, and between items judged to be relevant to the query and those not relevant. A typical contingency table is presented in Table 1(a), and four common evaluation measures derived from it are contained in Table 1(b).

Each of the measures listed in Table 1 is initially defined for each query separately. However, procedures exist for averaging the measures over a complete query set and for suitably displaying the resulting values in the form of recall-precision, or recall-fallout graphs. [6] These graphs are then expected to reflect the performance of an entire system for a given set of users.

It should be noted that the four retrieval measures are not

	Relevant	Not Relevant	
Retrieved	a	b	a+b
Not Retrieved	c	d	c+d
	a+c	b+d	a+b+c+d

(a) Contingency Table

Symbol	Evaluation Measure	Formula	Explanation
R	Recall	$\frac{a}{a+c}$	proportion of relevant actually retrieved
P	Precision	$\frac{a}{a+b}$	proportion of retrieved actually relevant
F	Fallout	$\frac{b}{b+d}$	proportion of nonrelevant actually retrieved
G	Generality	$\frac{a+c}{a+b+c+d}$	proportion of relevant per query

(b) Principal Evaluation Measures

Retrieval Evaluation Measures

Table 1

independent of each other. Specifically, three of the measures will automatically determine the fourth. As an example, equation (1) can be used to derive precision in terms of recall, fallout, and generality, as follows:

$$P = \frac{R \cdot G}{(R \cdot G) + F(1-G)} \qquad (1)$$

Most of the retrieval evaluation results published in the literature have been presented in terms of recall and precision. Since recall provides an indication of the proportion of relevant actually obtained as a result of a search, while precision is a measure of the efficiency with which these relevant are retrieved, a recall-precision output is user-oriented, in the sense that the user is normally interested in optimizing the retrieval of relevant items. On the other hand, fallout is a measure of the efficiency of rejecting the nonrelevant items, and includes as a factor the total number of nonrelevant in the collection (which in many cases is approximately equivalent to the collection size). For this reason, a recall-fallout display is normally considered to be systems-oriented since it indicates how well the nonrelevant are rejected as a function of collection size.

In view of their special orientation, it would then appear that some of the measures are more appropriate in certain circumstances than in others: in particular, if a systems viewpoint is important which takes into account the amount of work devoted to the retrieval of nonrelevant items as well as the collection size, a fallout display may be more desirable than a graph based on precision.

The situation is unfortunately complicated by the fact that the various measures do not vary in the same manner when a comparison is made of the performance of several distinct document collections. Consider, as as example, the parameter variations produced by changes in collection generality. As the generality increases, that is, as the average number of relevant per query grows larger, the number of relevant retrieved may also be expected to increase. In terms of the variables introduced in Table 1, a and a+c may then be expected to grow directly with generality; on the other hand a+b, and b+d (the total retrieved, and the total non-relevant) remain relatively constant.

As G increases, the following picture then emerges for R, P, and F, respectively:

$$R = \frac{\uparrow}{\uparrow}, \quad P = \frac{\uparrow}{\rightarrow}, \quad F = \frac{\rightarrow}{\rightarrow}$$

where the upward arrow denotes an increasing quantity, and the horizontal arrow a quantity more or less constant. Thus, R and F should remain reasonably constant with changes in generality, since numerator and denominator vary in the same direction. Precision, on the other hand, should vary directly with generality because of the increasing numerator together with the constant denominator.

This kind of argument has been used in the past to show that the use of recall-precision graphs is generally undesirable, [7], and to claim that performance figures obtained with small sample collections in a laboratory environment cannot be applied to large operational collections [8]. This question is further examined in the next section.

3. Variations in Collection Size

A) Theoretical Considerations

Consider a performance comparison for two collections of different size within a given subject area. Such collections generally exhibit different generality characteristics, since the larger collection is likely to contain on the average many more nonrelevant items per query, and therefore proportionately many fewer relevant ones.

In going from the smaller (test) collection to the larger (operational) one, two limiting cases may be distinguished:

a) if the relevance of the documents added to the small collection in order to produce the large one is difficult to assess in a clear-cut way, and nonrelevant items that are hard or easy to reject are added roughly in the same proportion as originally present, then for a given level of recall a larger number of relevant items will have to be retrieved; this will imply the simultaneous retrieval of a larger number of nonrelevant, thereby depressing precision, but keeping fallout roughly constant;

b) on the other hand, if the documents added are clearly extraneous to the query topics and the nonrelevant ones are easily rejectable, the number of relevant and nonrelevant retrieved at a given recall level remains constant, thereby producing a constant precision but lower fallout for the larger collection; the situation is summarized in Table 2.

If case 2 were to occur in practice, that is, if one could insure that any nonrelevant documents added to the small collection would be

	Large Collection	
Small Collection	① Addition of Partly Relevant and Non-relevant in same Proportion	② Addition of Extraneous Clearly Non-relevant
P	P ↓	P →
F	F →	F ↓

Precision and Fallout Performance for Variations in Collection Size

Table 2

easy to reject, then the standard recall-precision plot would furnish a completely adequate evaluation tool, since the precision would then be independent of the generality change, and would in fact be identical for both collections at each common recall level. If, on the other hand, case 1 is taken as typical, then fallout can be assumed to be constant. This makes it possible to compute an "adjusted precision" value as a function of generality, to account for the generality change in upgrading from a small collection to a large one.

Consider, as an example, a document collection with generality G_1, and a given precision P_1 at a recall level of R_1. If the size of the collection is altered to a new generality G_2, then, for any given recall level, equation (1) can be used to compute the adjusted precision P_2 for the larger collection. In fact, if the generality change is subject to the rules of case 1, one has (from equation (1)):

$$P_2 \text{ (adjusted)} = \frac{R_1 \cdot G_2}{(R_1 \cdot G_2) + F_1(1-G_2)} \qquad (2)$$

where the computations are made for a given recall level $R_1 = R_2$, and fallout is assumed constant. Equation (2) then provides a means for computing the precision transformation for the case where all factors other than generality remain constant.

Cleverdon and Keen propose a three-step procedure for effecting the precision transformation as follows: [1]

a) given G_1, R_1 and P_1 compute F_1;

b) assume $F_1 = F_2$;

c) given G_2, $R_1 = R_2$, and F_1, compute P_2.

An example for a collection of generality 0.005 and recall and precision values of 0.60 and 0.25 respectively is shown in Table 3. The precision adjusted to a generality level of G = 0.002 is seen to be 0.11.

B) Evaluation Results

The theoretical considerations outlined in the last few paragraphs indicate that the retrieval evaluation provides an accurate picture for the case where the expansion in collection size is caused by the addition to a small document collection of clearly nonrelevant items which are easily rejectable, and for the case where fallout remains constant, that is, where relevant and nonrelevant items are added in a proportion roughly equivalent to that which originally existed.

Unfortunately, when the assumptions of cases 1 and 2 are tested on actual document collections of different generality, they are found not to hold in practice. For example, in a test conducted some years ago with two document collections of 200 and 1400 documents in aerodynamics, respectively, and a sample of 42 queries, Cleverdon and Keen found for a specified cutoff and processing method that

> "b (the nonrelevant retrieved) has increased by a factor of 5.2352 while the total number of nonrelevant documents in the collection (b+d) has increased by a factor of 7.1448." [1, p.74]

For the example considered, fallout therefore did not remain constant, and many of the nonrelevant included in the larger collection of 1400 items obviously exhibited a lower probability of being retrieved than the nonrelevant included in the smaller subcollection.

	Large Collection	
Small Collection	① Addition of Partly Relevant and Nonrelevant in same Proportion	② Addition of Extraneous Clearly Nonrelevant
P	P ↓	P →
F	F →	F ↓

Precision and Fallout Performance for Variations in Collection Size

Table 2

Parameter		Collection 1		Collection 2
G		.005		.002
R		.60		.60
P	step 1	.25	step 3	.11 (adjusted P)
F		.00905		.00905
			step 2	

Precision Transformation for Constant Fallout

Table 3

To verify this result, the two collections originally used by Cleverdon were subjected to a complete retrieval test, using a set of 36 queries with identical relevance properties in both collections (the set of relevant items was the same for each query in both collections). The collection characteristics are summarized in Table 4, and recall-precision, as well as recall-fallout, plots are included in Fig. 1, averaged over the 36 test queries. [9]

It may be seen from the output of Table 4 and Fig. 1 that although the collection generality decreases by a factor of about seven in the transition from small to large collection, the fallout decreases by a factor of only three on the average. Thus the proportion of nonrelevant retrieved is much smaller for the large collection than for the small one, producing the recall-fallout plot of Fig. 1(b) which favors the large collection (the smaller the fallout, the better is the performance).* The recall-precision plot, on the other hand, favors the small collection (the higher the precision, the better is the performance), indicating that at a given recall level, fewer nonrelevant will have been retrieved for the small collection than for the large one.

The data of Table 5, containing the average number of nonrelevant documents retrieved at various recall levels, indicate that the seven-fold decrease in collection generality is accompanied by an increase in the average number of nonrelevant retrieved, ranging from a factor of 2 at a recall of 0.1 to a factor of only 3.2 at a recall of 0.3 and 0.5. This explains the superior systems-oriented performance of the large 1400 collection in comparison with the small one.

---

*The average number of nonrelevant items retrieved at various recall levels is shown in Table 5 for the Cranfield 200 and 1400 collections.

Property	Cranfield 200	Cranfield 1400
Source	Cranfield document abstracts in aerodynamics	Cranfield document abstracts in aerodynamics
Document Analysis	Word stem process	Word stem process
Number of Documents	200	1400
Number of Queries	36	36
Number of Relevant Documents	160	160
Type of Search	Full search	Full search
Generality	.0222	.0031
Average Fallout	.0248	.0081

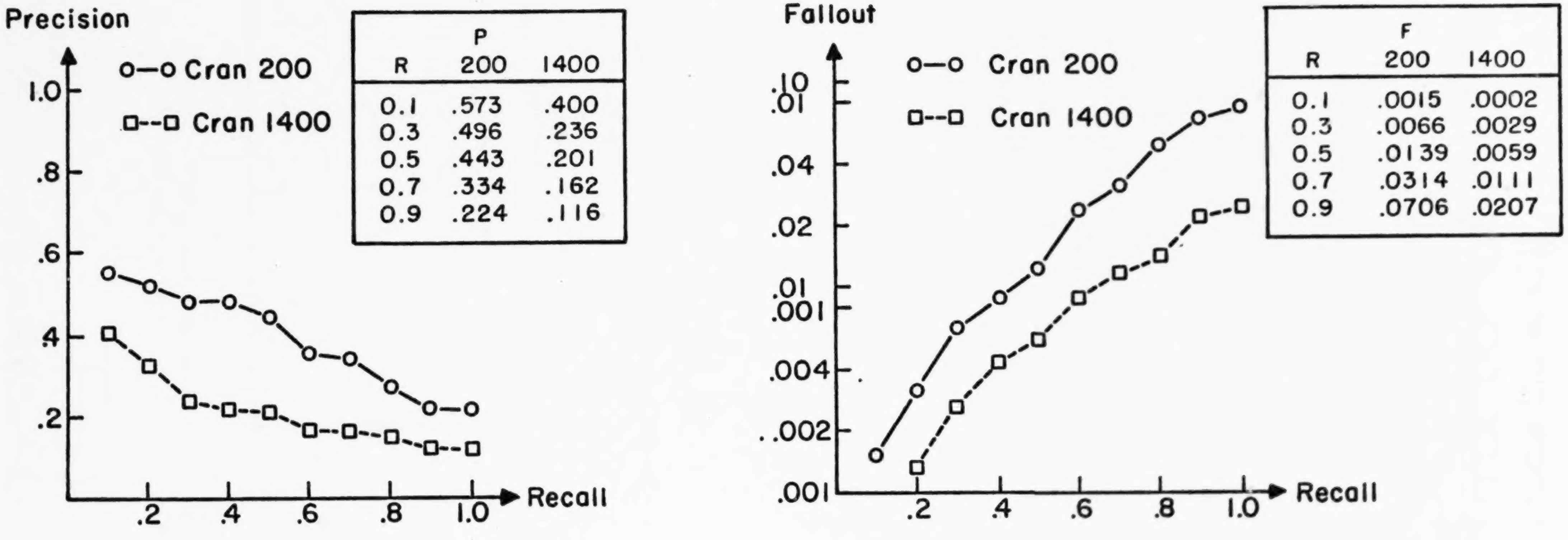
Collection Properties for Cranfield 200 and 1400

Table 4

Recall	Average Number of Nonrelevant Retrieved		Factor of Increase from 200 to 1400
	Cranfield 200	Cranfield 1400	
0.1	0.33	0.67	2
0.3	1.35	4.32	3.2
0.5	2.79	8.82	3.2
0.7	6.21	16.15	2.6
0.9	13.89	30.54	2.2

Increase in Nonrelevant Retrieved
from Cranfield 200 to Cranfield 1400

Table 5



R	P 200	P 1400
0.1	.573	.400
0.3	.496	.236
0.5	.443	.201
0.7	.334	.162
0.9	.224	.116

R	F 200	F 1400
0.1	.0015	.0002
0.3	.0066	.0029
0.5	.0139	.0059
0.7	.0314	.0111
0.9	.0706	.0207

a) Recall-Precision

b) Recall-Fallout

Performance Comparison for Cran 200 and Cran 1400 Collections
(averages over 36 queries; word stem process)

Fig. 1

In practice, it is seen that the larger the collection (and therefore the smaller the generality), the larger will be the number of nonrelevant items which will have been retrieved at any given recall level; however the resulting decrease in precision performance is much smaller than expected by the factor of increase in collection size and nonrelevant items added. Neither of the two simple generality transformations discussed in the preceding subsection appears to be applicable in practice, since both precision and fallout may be expected to decrease with a decrease in collection generality.*

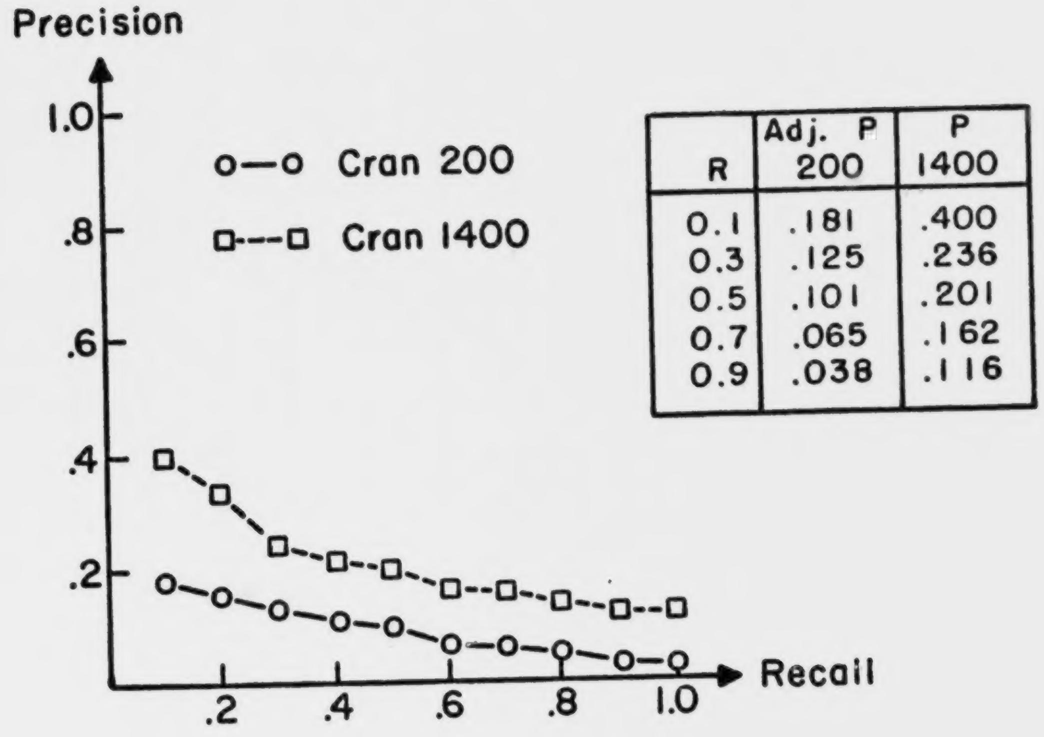
C) Feedback Performance

It is known that interactive search methods in which the user influences the retrieval process by providing appropriate feedback information during the course of the operations can be used profitably in a retrieval environment. [10,11] In fact, some of the feedback methods which have been tested over the last few years, including, in particular, the relevance feedback process regularly used with the automatic SMART document retrieval system, provide anywhere from five to twenty percent improvement in precision at a given recall level. Most other refinements in retrieval methodology — such as, for example, a particularly sophisticated language analysis scheme — may bring improvements in performance of the order of a few percent at best.

The relevance feedback process utilizes user relevance judgments

---

*If the precision transformation of equation (1) were (incorrectly) to be applied to the precision performance of the small collection to reduce its generality to that of the large collection (.0031), the adjusted precision curve of Fig. 2 would result. This adjusted precision is an inverse function of fallout, which accounts for its inferior performance compared with that of the large collection.



R	Adj. P 200	P 1400
0.1	.181	.400
0.3	.125	.236
0.5	.101	.201
0.7	.065	.162
0.9	.038	.116

Recall-Precision Plot for Cran 200 and Cran 1400 Collections
(Precision Adjusted to Generality of .0031)
(averages over 36 queries)

Fig. 2

for documents previously retrieved by an initial search in order to construct an improved query formulation which can subsequently be used in a new "first iteration", or "second iteration" search. Specifically, an initial search is performed for each request received, and a small amount of output, consisting of some of the highest scoring documents, is presented to the user. Some of the retrieved output is then examined by the user who identifies each document as being either relevant (R) or not relevant (N) to his purpose. These relevance judgments are later returned to the system, and used automatically to adjust the initial search request in such a way that query terms present in the relevant documents are promoted (by increasing their weight), whereas terms occurring in the documents designated as nonrelevant are similarly demoted. This process produces an altered search request which may be expected to exhibit greater similarity with the relevant document subset, and greater dissimilarity with the nonrelevant set.

The altered request can next be submitted to the system, and a second search can be performed using the new request formulation. If the system performs as expected, additional relevant material may then be retrieved, or, in any case, the relevant items may produce a greater similarity with the altered request that with the original. The newly retrieved items can again be examined by the user, and new relevance assessments can be used to obtain a second refoumulation of the request. This process can be continued over several iterations, until such time as the user is satisfied with the results obtained.

In order to determine whether the relevance feedback process is usable with large document collections in an operational environment, the feedback procedure was tested using two collections in aerodynamics of different generality. [12] If comparable feedback improvements could be obtained for collections of varying size and generality, then it would appear reasonable to conclude that the feedback process will be valuable under operational conditions.

The two collections being tested consist of 200 and 424 document abstracts in aerodynamics, respectively, together with 22 queries with identical relevance properties in both collections. The collection characteristics are summarized in Table 6, and the recall-precision and recall-fallout graphs obtained with a "positive" feedback strategy are shown for both collections if Figs. 3 and 4.

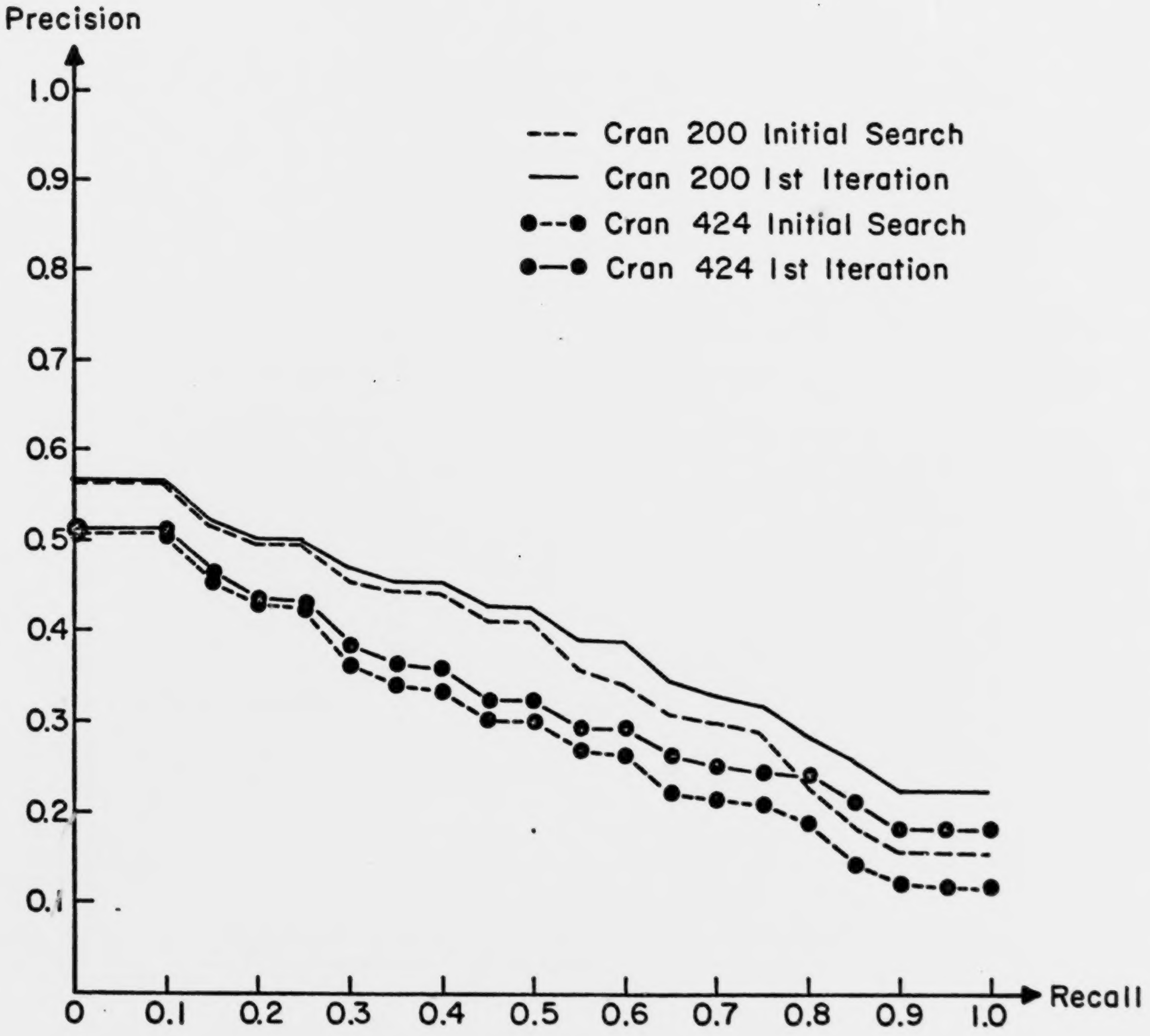
It may be noted that once again the recall-precision output favors the small collection, whereas the recall-fallout output is more favorable to the larger collection. Furthermore, while the generality decreases by a factor of over 2 from small to large collection, the fallout drops by less than one-half. These results are entirely in agreement with those previously obtained for the Cranfield 1400 collection. The output of Figs. 3 and 4 for the positive feedback strategy also indicates that the magnitude of improvement provided by one feedback iteration is approximately comparable for the two collections.

In order to investigate the question of feedback improvement in more detail, several feedback procedures were tested including, in particular, the following three types (based on the retrieval of the top five documents in each case):

Property	Cranfield 200	Cranfield 424
Source	Abstracts in aerodynamics	Abstracts in aerodynamics
Analysis	Word stem process	Word stem process
No. Documents	200	424
No. Queries	22	22
No. of Relevant	115	115
Search	Feedback search	Feedback search
Generality	.0261	.0123
Ave. Fallout	.0333	.0211

Collection Properties for Feedback Searches
Using Cranfield 200 and 424

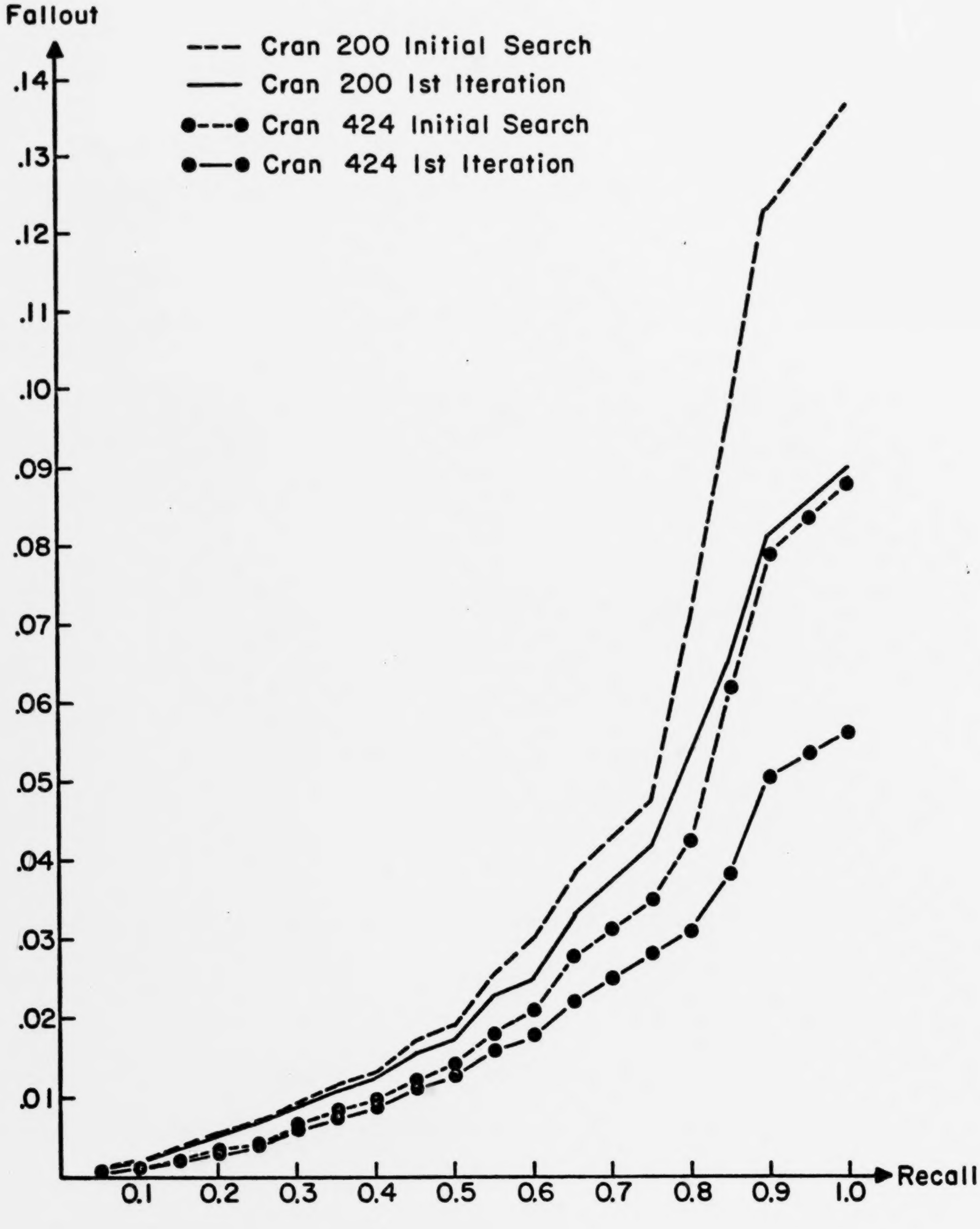
Table 6



Recall-Precision Comparison for Cran 200 and 424 Collections

(initial run and one feedback iteration—positive feedback only, word stem process, 22 queries)

Fig. 3



Recall-Fallout Graph for Cran 200 and 424
(initial run and one feedback iteration-
positive feedback only)

Fig. 4

a) positive feedback, where information obtained from documents known to be relevant is used to update the query formulation;

b) selective negative feedback, where positive information is derived from the relevant documents together with negative information obtained from the top retrieved nonrelevant item;

c) modified selective negative feedback, where the negative information derived from the nonrelevant documents is used only when no positive information is available.

The evaluation is based principally on two evaluation functions, which measure respectively the precision improvement and the fallout improvement as follows: [12]

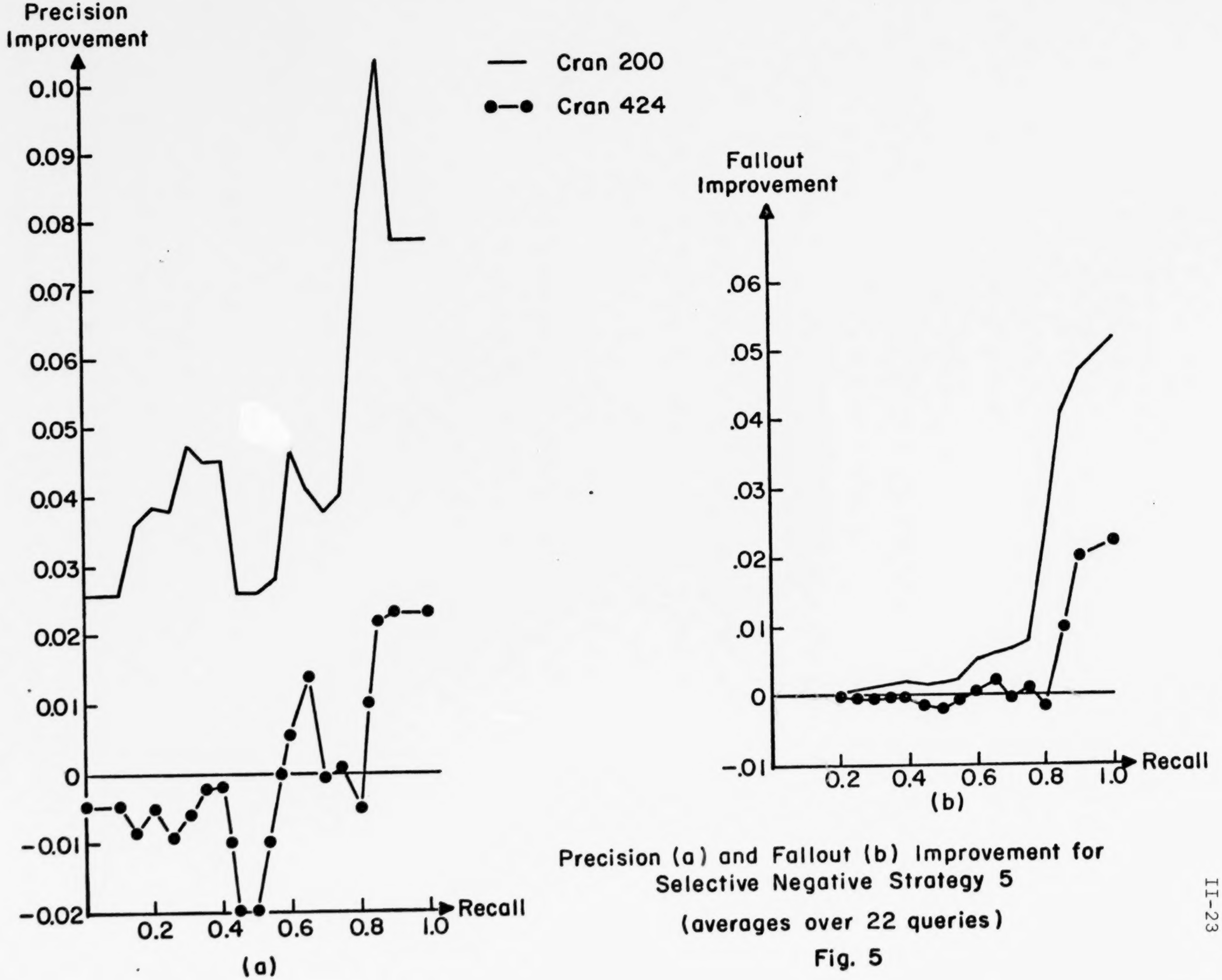
$$\text{Precision improvement} = P_1 - P_0 \,,$$

where P_0 is the precision of the initial search, and P_1 is the precision of the feedback iteration at a specified fixed recall point; and

$$\text{Fallout improvement} = F_0 - F_1 \,,$$

where F_0 is initial fallout, and F_1 the fallout of the feedback iteration. (A performance improvement implies that the fallout for the feedback iteration is smaller than the initial fallout.)

The output for a selective negative feedback strategy which does not operate satisfactorily in an environment of decreasing generality is shown in Fig. 5. It is seen that for the larger collection the precision improvement is negative for most recall points, showing that the feedback process in fact hurts the performance. The same is true for some points of the fallout improvement curve. Apparently, the strategy represented by



Precision (a) and Fallout (b) Improvement for Selective Negative Strategy 5

(averages over 22 queries)

Fig. 5

the curves of Fig. 5 uses too many nonrelevant items for feedback purposes thereby hurting retrieval. (Fewer relevant items are retrieved early in the search for the Cran 424 collection, than for Cran 200.)

The performance for two feedback strategies which operate excellently with decreases in generality is shown by the precision and fallout improvement curves of Figs. 6 and 7. Fig. 6 covers the positive-feedback strategy which is seen to operate equally well for both collections. Still larger improvements are noted in Fig. 7 for the modified negative strategy in which a nonrelevant item is used for feedback purposes only when positive information (in the form of relevant retrieved documents) is not available.

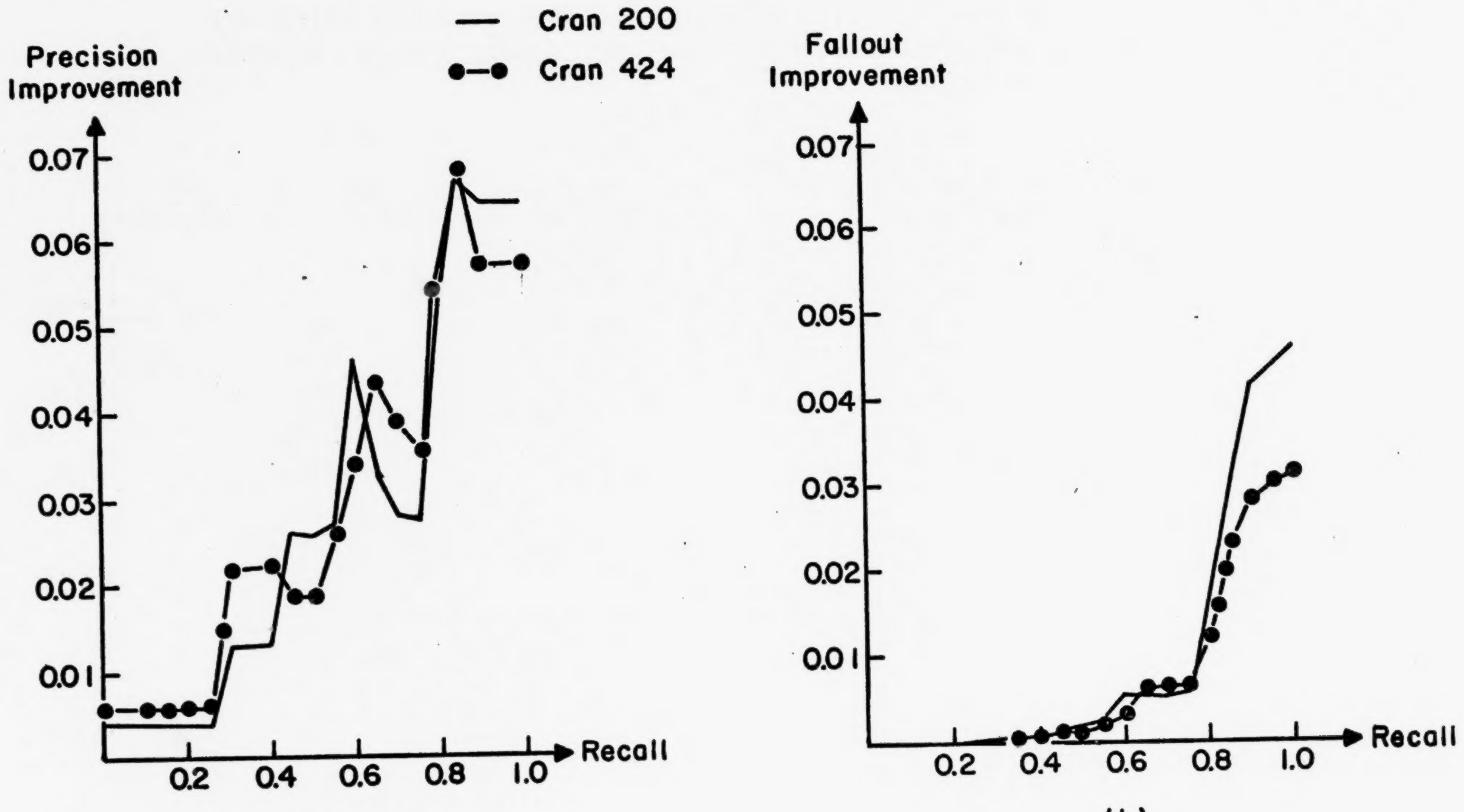
From the output of Figs. 6 and 7 it appears that feedback strategies can be implemented which operate equally well for collections of low and high generality. These strategies should be implementable in a realistic environment comprising thousands of items where they may be expected to produce the performance improvements previously noted for small test collections.

4. Variations in Relevance Judgments

A generality problem arises not only when collections of different size but identical relevance properties are to be compared, but also when the same collection is processed with different types of relevance assessments. In a previous study, a collection of 1268 documents in library science and documentation was examined using four types of relevance grades:

a) the A judgments representing relevance assessments by the query authors;

b) the B judgments representing nonauthor judges;



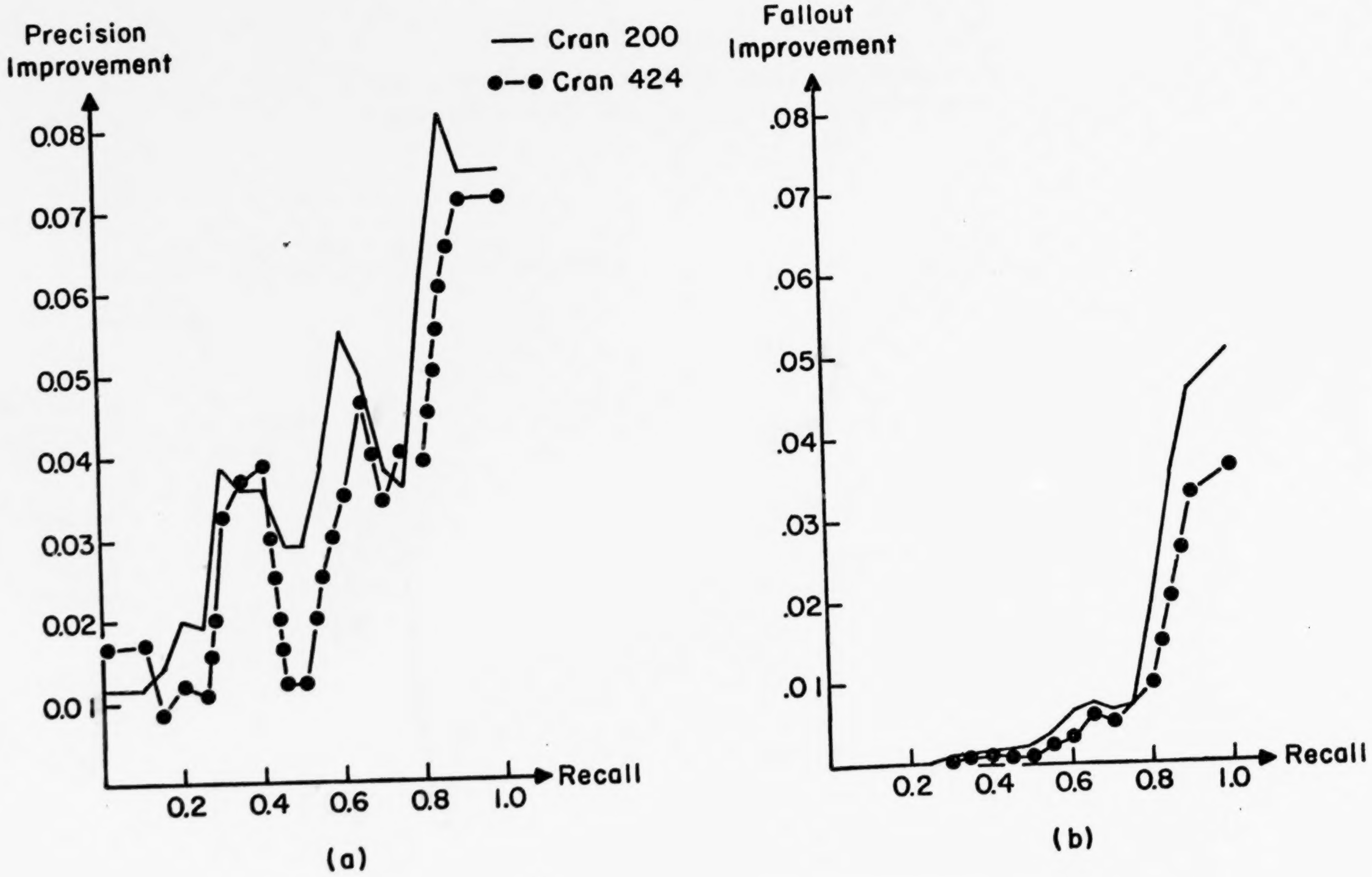
(a)

(b)

Precision (a) and Fallout (b) Improvement for Feedback Strategy I (positive feedback)

(averages over 22 queries)

Fig. 6



Precision (a) and Fallout (b) Improvement for Feedback Strategy 4
(modified selective negative strategy; averages over 22 queries)

Fig. 7

c) the C judgments representing the disjunction between the A and B judgments (that is, a document is judged relevant to a query if either A or B judges termed it relevant);

d) the D judgments representing the conjunction between A and B judgments (a document is judged relevant if both A and B judges termed it relevant).

It was demonstrated in the previous study [13], that the recall-precision performance graphs are relatively invariant to the variations caused by the multiple relevance assessments, and by the resulting changes in generality.

In an attempt to determine whether the performance characteristics obtained with collections of different size can be related to those produced by collections with varying relevance properties, the C and D collections are processed once again under slightly modified conditions. The collection properties are outlined in Table 7.

It will be noted that in the present case the generality change is produced not by adding any documents to the C collection in order to obtain the other collection of lower generality, but rather by subtracting from the set of relevant documents a number of items about which a unanimity of opinion could not be obtained by the relevance assessors. Nevertheless, the performance figures given in Table 7, and in Fig. 8(a) show that once again somewhat better recall-precision data for the collection of high generality (the C collection) are coupled with somewhat better fallout data for the collection of low generality (the D collection).*
This reflects the fact, on the one hand, that precision varies somewhat

---

*The recall-precision figures shown in Fig. 8(a) are not directly comparable to those produced in the earlier study [13] because of a small difference in the method used to produce performance averages over the total number of queries.

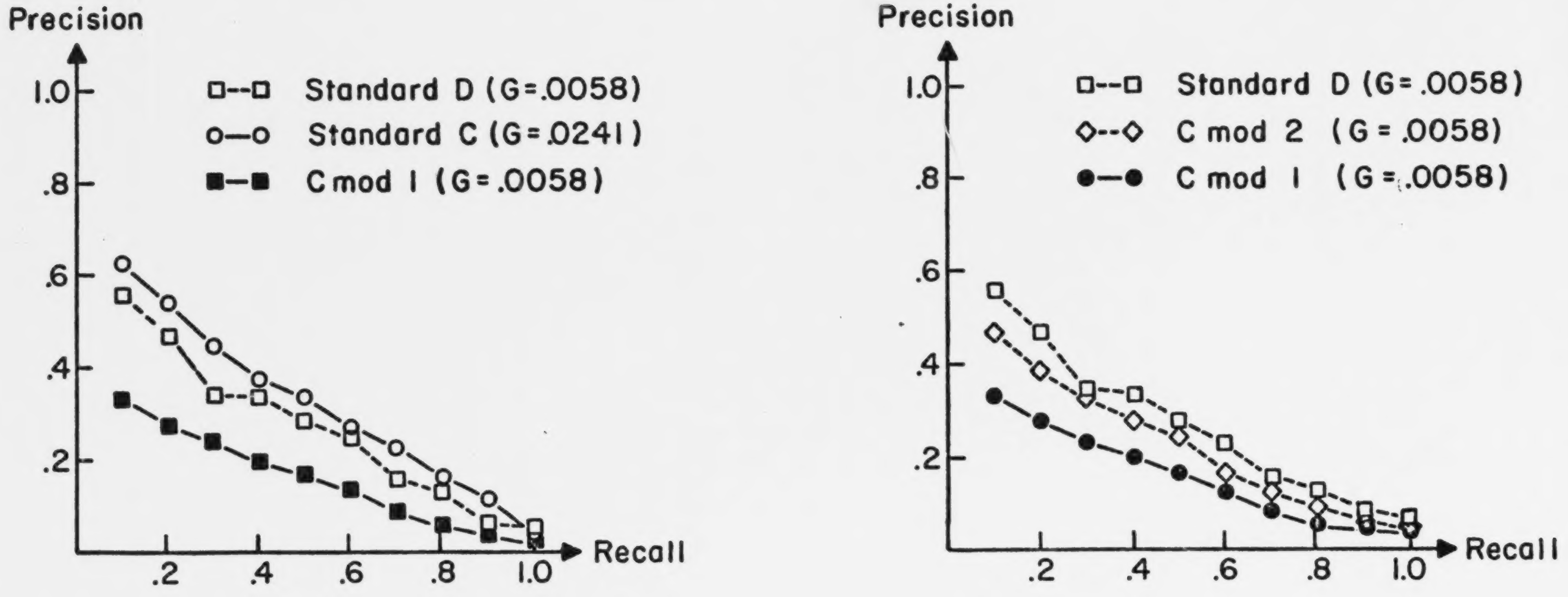
Property	Ispra C	Ispra D
Source	Document abstracts in documentation	Document abstracts in documentation
Analysis	Thesaurus	Thesaurus
No. Documents	1268	1268
No. Queries	45	45
No. of Relevant	1260	306
Search	Full search	Full search
Generality	.0241	.0058
Average Fallout	.1409	.0819

Collection Properties for Ispra C and D Collections

Table 7

	P		
R	D	C	C mod 1
0.1	.579	.627	.343
0.3	.344	.447	.234
0.5	.291	.334	.177
0.7	.155	.213	.099
0.9	.079	.111	.055

	P		
R	D	C mod 2	C mod 1
0.1	.579	.481	.343
0.3	.344	.332	.234
0.5	.291	.251	.177
0.7	.155	.127	.099
0.9	.079	.057	.055



a) Comparison of Standard C and D

b) Comparison of Modified C Runs

Generality Comparison for Collections of Fixed Size

C mod 1 : $n_1 - k$ relevant items randomly set nonrelevant

C mod 2 : n_2 relevant retained; remainder scattered

Fig. 8

with generality, and therefore the collection with higher generality is likely to produce better precision. On the other hand, the collection of low generality exhibits better relevance judgments, since at least two judges had to agree on the relevance of each document; there exists therefore a greater certainty about the relevance (or nonrelevance) of each document with respect to each query, which implies that the nonrelevant are easier to reject using the D relevance judgments.

In order to see how the performance data change under a generality transformation, the C collection with high generality (.0241) is reduced to the generality of the D collection (.0058) in two different ways:

a) collection C mod 1 is produced by taking 962 relevant documents chosen at random and calling them nonrelevant; this reduces the original set of 1260 relevant documents in C to a total of 306 relevant (equal to the number of relevant in D);

b) collection C mod 2 is produced by retaining 306 out of the 1260 originally relevant items; the remaining 962 formerly relevant items are assigned random ranks in the collection instead of being retained with the rank they initially possessed as in C mod 1).*

The performance of the modified C collections which now exhibit the same generality as the standard D is presented in the recall-precision graphs of Fig. 8(b). It is seen that when the generality is kept invariant, as it is for the three collections of Fig. 8(b), the collection with the most reliable relevance judgments (the standard D) produces the best performance. Of the two modified C collections obtained by the generality transformation, the second produces better output than the first, since it is more carefully constructed by randomly deleting relevant items, and then randomly reintroducing them as nonrelevant ones with new ranks.

---

*The reranking process followed is described in a note by Williamson. [14]

5. Summary

A variety of retrieval tests were performed with collections of varying generality in the areas of aerodynamics and documentation. Since precision varies with generality, the precision output generally favors the (small) collection of high generality. However, as the generality drops by a factor of k, the precision drops by a much smaller factor, and the fallout, which had been thought to remain invariant with generality changes, in fact decreases with generality, and thus favors the (large) low generality collections.

No clear extrapolation appears possible at this time which would permit a prediction to be made about the likely performance of very large collections of several hundred thousand items. However, the fallout data obtained in this study make it clear, that an argumentation which claims that the retrieval of 20 nonrelevant items for a collection of 1000 items would necessarily lead to an expected retrieval of 20,000 nonrelevant for a collection of a million is fallacious, since it assumes a constant fallout performance.

The user feedback procedures appear to be useful for collections of varying generality, and they should be implemented in operational environments. Finally, when generality variations arise from inconsistencies in the relevance assessments, the collection with the most secure relevance data performs best.

As larger document collections come into experimental use, the fallout and precision figures should continue to be compared with the generality variations. In this fashion, it may be possible, in time, to obtain reliable projections for the performance with large collections under operational conditions.

References

[1] C. W. Cleverdon and E. M. Keen, Factors Determining the Performance of Indexing Systems, Vol. 2, Test Results, Aslib Cranfield Research Project, Cranfield, 1966.

[2] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill Book Company, New York, 1968.

[3] G. Salton and M. E. Lesk, Computer Evaluation of Indexing and Text P ocessing, Journal of the ACM, Vol. 15, No. 1, January 1968.

[4] G. Salton, Automatic Processing of Foreign Language Documents, Journal of the ASIS, Vol. 21, No. 3, May-June 1970.

[5] G. Salton, A Comparison between Manual and Automatic Indexing Methods, American Documentation, Vol. 20, No. 1, January 1969.

[6] E. M. Keen, Evaluation Parameters, Report No. ISR-13 to the National Science Foundation, Section II, Department of Computer Science, Cornell University, January 1968.

[7] S. E. Robertson, The Parametric Description of Retrieval Tests — Part 1: Basic Parameters, Journal of Documentation, Vol. 25, No. 1, March 1969.

[8] D. R. Swanson, Implications of the SMART-Medlars Comparison for Very Large Collections, unpublished manuscript, University of Chicago, 1970.

[9] R. G. Crawford, A Generality Study, Project Report, Computer Science Department, Cornell University, January 1970.

[10] G. Salton, Search and Retrieval Methods in Real-Time Information Retrieval, Proc. IFIP Congress 68, North Holland Publishing Company, Amsterdam 1969, p. 1082-1093.

[11] M. E. Lesk and G. Salton, Interactive Search and Retrieval Methods Using Automatic Information Displays, Proc. AFIPS Spring Joint Computer Conference, AFIPS Press, Montvale, N. J., 1969, p. 435-446.

[12] B. Capps and M. Yin, Effectiveness of Feedback Strategies on Collections of Differing Generality, Scientific Report ISR-18, Department of Computer Science, Cornell University, October 1970.

[13] M. E. Lesk and G. Salton, Relevance Assessments and Retrieval System Evaluation, Information Storage and Retrieval, Vol. 4, No. 4, 1969, p. 343-359.

[14] R. Williamson, A Proposal for an Experiment to Ascertain the Relationship between the Generality Ratio and Performance Measures, unpublished notes, Cornell University, 1969.

III. Automatic Indexing Using Bibliographic Citations

G. Salton

Abstract

Bibliographic citations attached to technical documents have been used variously to refer to related items in the literature, to confer importance to a given piece of writing, and to serve as supplementary indications of document content. In the present study, citations are used directly to identify document content, and an attempt is made to evaluate their effectiveness in a retrieval environment. It is shown that the use of bibliographic citations in addition to the normal keyword-type indicators produces improved retrieval performance, and that in some circumstances, citations are more effective for retrieval purposes than other more conventional terms and concepts.

1. Significance of Bibliographic Citations

The role of bibliographic citations attached to scientific and technical documents has received intensive study for many years. Several authors have noted, in particular, that the number of incoming citations (that is, the number of citations from a given set of outside documents to a specified target document) constitute useful indicators of document type and importance [1,2]. In consequence, the so-called "bibliographic network" consisting of documents and citations between them has been used to assess the characteristics of scientific and technical communications. [3]

In addition to providing indications of document influence, bibliographic citations also play a role as content identifiers. The close affinity between the citations attached to a given document and the normal keyword-type content indicators has been expressed by Garfield in the following terms [4]:

> "By using the author's references in compiling the citation index, we are in reality using an army of indexers, for every time an author makes a reference, he is in effect indexing that document from his point of view...."

Furthermore, only a very small proportion of documents appears to be totally disconnected from the bibliographic network, in the sense that these documents do not cite any other documents nor are they cited from the outside[3]:

> "...there is a lower bound of one percent of all papers that are totally disconnected in a pure citation network, and could be found only by topic indexing...."

As a result, search tools such as the "citation index" which lists all incoming citations for each document in the index have proved to be useful adjuncts to information search and retrieval.

A variety of studies have been undertaken in an attempt to determine the relationship between standard keywords and bibliographic citations for content analysis purposes. Thus, it was determined that papers which were related by similarities in bibliographic citation patterns also provided a large number of common subject identifiers. [5] Furthermore, the correlation between citation similarities on the one hand, and index term similarities on the other is found to be far greater than expected for random document sets. [6]

While bibliographic citations appear not to have been used directly as content indicators for retrieval purposes up to the present time, a number of experiments have been performed in which citations were incorporated as feedback information during the search process, in an attempt at retrieving additional information similar to that being identified in the search. [7,8] Specifically, an initial search would be made, leading to the retrieval of a number of documents. These would be scanned by the user, and information about these documents — including in particular document authors, citations made by the documents, and authors of these citations — would be returned to the system to be incorporated into an improved search formulation. The evaluation of this bibliographic feedback process proved, in particular, that [8]:

> "...no differences greater than four percent were found between the results of feeding back only subject data, and those of feeding back only bibliographic data. This implies that the usefulness of bibliographic data for feedback is of the same order as that of subject descriptors."

In addition, the same study showed that when citation data were added to standard subject indicators in a feedback environment, improvements of up to ten percent in retrieval effectiveness were obtained over and above the results produced by subject information alone. This led to the conjecture that [8]:

> "Since the bibliographic information is useful for feedback purposes, it should also prove valuable for initial retrieval searches."

An attempt is made in the remainder of this study to evaluate the correctness of this statement. Specifically, a collection of 200 documents in the field of aerodynamics is processed against a set of 42 queries using first the normal content analysis methods incorporated into the automatic SMART document retrieval system [9], and then a modified process based on the bibliographic citations attached to the documents. The test design and evaluation results are covered in the remaining sections of this report.

2. The Citation Test

Consider a given document collection available in the form of English language abstracts, together with a corresponding set of user queries. Given such a collection, various linguistic analysis procedures may serve to reduce each item into analyzed vector form. A concept vector, representing either a document or a query, normally consists of a set of terms, or concepts, together with the respective concept weights. Two of the content analysis methods most frequently used with the SMART retrieval system are the word stem, and the thesaurus processes. In a word stem analysis, each concept incorporated into a normal concept vector represents a word stem extracted from the document, whereas for the thesaurus procedure, the concepts represent thesaurus categories obtained by consulting an automatic dictionary during the analysis operations. Word stems, or thesaurus categories are then concepts somewhat similar to the standard subject indicators normally assigned manually to queries and documents. In such an environment, the normal retrieval operation would consist in matching the concept vectors for queries and documents, and in retrieving for the users' attention all documents whose vectors exhibit a reasonable degree of similarity with the corresponding query vectors.

If it is assumed that each document carries with it a set of bibliographic citations (either to or from the document), it is possible to add to the normal document concept vectors, suitably chosen codes representing the bibliographic citations; alternatively, the citation codes might replace the normal concepts.

In order to obtain a match between citation codes attached to documents and normal user queries, it becomes necessary to attach citation information also to the queries. This can be done in one of two ways:

a) some queries may have been formulated by the user population in response to a set of documents known in advance to be relevant; that is, for each query one or more source documents exist, and the user's query is designed to retrieve additional items similar to the respective source documents*;

b) alternatively, a source document does not exist in advance, but the user is able to designate some other document as likely to be relevant to his query.

In either case, it becomes possible to add to the query vectors citation codes corresponding to source document citations, or to citations attached to the designated relevant documents, as the case may be.

These operations then produce expanded query and document vectors consisting partly of standard concept codes, and partly of citation codes, as shown schematically in Figure 1. Three types of retrieval operations become possible:

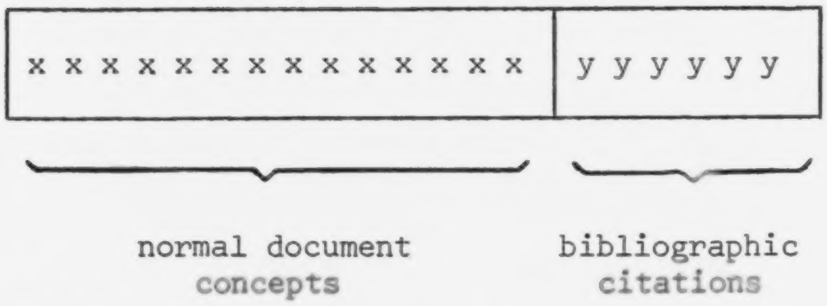
a) using only standard subject identifiers (the 'x' concepts of Figure 1);

b) using only citation concepts (the 'y' concepts of Figure 1);

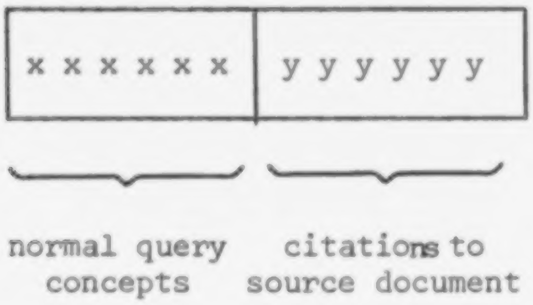
c) using both the standard and the citation concepts (the 'x'

---

*In a previous test in which original query formulations were replaced by source document vectors, it was shown that the retrieval effectiveness produced by the source document "queries" was substantially better than that obtained with the standard queries. [10]


x x x x x x x x x x x x x x x
y y y y y y
normal document concepts
bibliographic citations

a) Typical Expanded Document Vector


x x x x x x
y y y y y y
normal query concepts
citations to source document

b) Typical Expanded Query Vector

Expanded Query and Document Vectors

Figure 1

and 'y' information).

In these circumstances, the relative value of the citation information may be ascertained by comparing the results obtained with these three types of concept vectors.

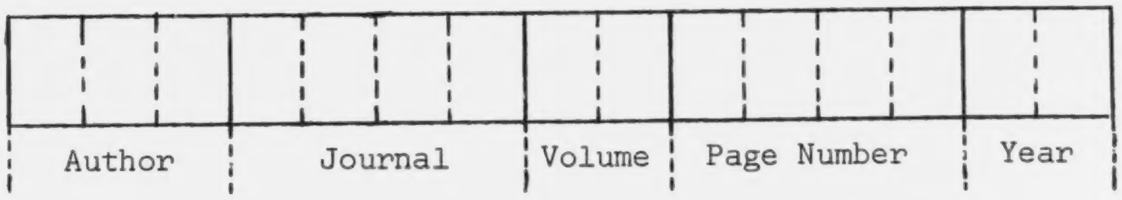
For the test under discussion, a collection of 200 document abstracts in aerodynamics was used with 42 search requests obtained from research workers in aerodynamics (the Cranfield collection [11]). Each document carried an average of 18 bibliographic references (outgoing citations to other documents), and each query was originally formulated in response to a source document. The set of source documents were similar in nature to the standard documents, in the sense that bibliographic citations were available for each; however, no source document was included among the standard 200.

To generate the citation portion of the document and query vectors, each citation was represented by a 15-character code. The citation coding is outlined in Figure 2, and some encoded sample documents are exhibited in the appendix. In order to increase the similarity coefficient for all documents cited by the query source documents, a citation code was added to each document vector not only for all outgoing citations, but also for each of the original documents. That is, each document is assumed implicitly to cite also itself (self-citation). A match between a query citation concept and a document citation concept may then be due to one of two causes:

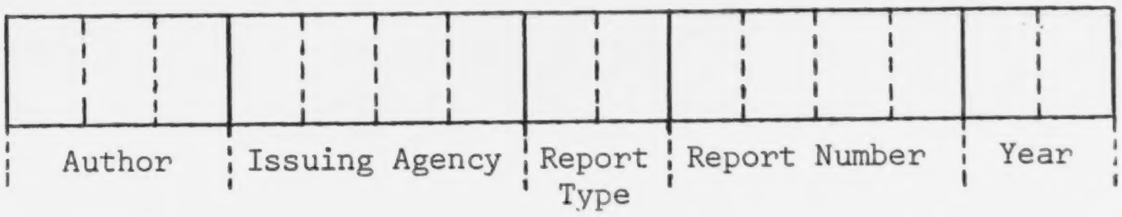
a) a request citation (source document citation) is identical with the document itself (request cites document);

b) a request citation is identical with a citation from a document (request and document have a common citation).

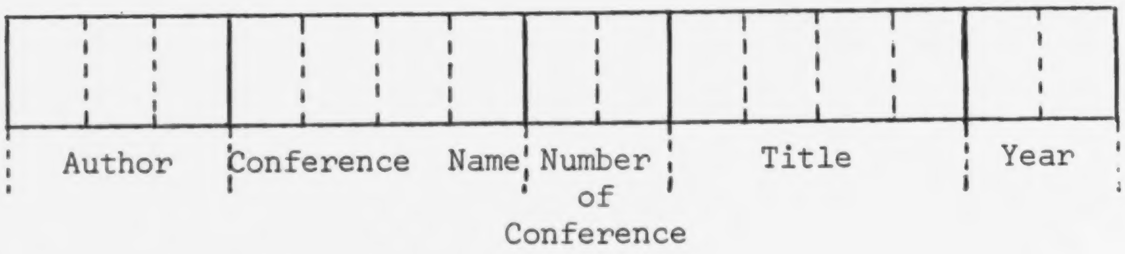
A comparison between citation effectiveness and standard concepts


Author
Journal
Volume
Page Number
Year

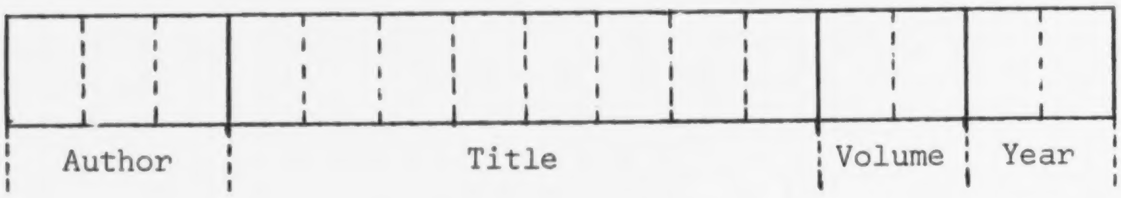
a) Typical Journal Code


Author
Issuing Agency
Report Type
Report Number
Year

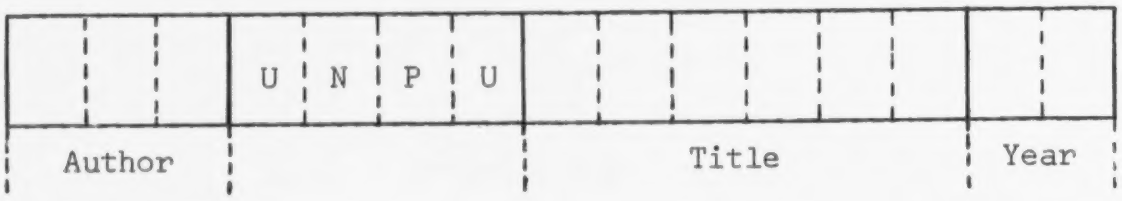
b) Typical Report Code


Author
Conference Name
Number of Conference
Title
Year

c) Typical Conference Paper Code


Author
Title
Volume
Year

d) Typical Book Code


U
N
P
U
Author
Title
Year

e) Unpublished Paper Code

Citation Coding

Figure 2

is obtained as usual by computing recall and precision values for the various runs while comparing the output.* The performance results are described in the remaining sections of this study.

3. Evaluation Results

The computation of recall and precision results depends on the availability of relevance assessments stating the relevance characteristics of each document with respect to each query. The original ("A") relevance assessments for the Cranfield collection were obtained by first submitting to the query authors for assessment the set of all documents cited by the source document, followed by additional items likely to be relevant. Since the source document citations were thus given special treatment, a bias may exist in favor of these citations — that is, an item cited by the source document may be more likely to be assessed as relevant than other extraneous documents. For this reason, three additional sets of relevance judgments were independently obtained from nonauthor subject experts, for which all documents were treated equally; that is, no special identification was provided for source document citations. The characteristics of the four sets of relevance assessments are summarized in Table 1.**

It may be seen that the four types of relevance assessments fall into two main categories as follows:

a) sets A and B have low generality characteristics — only four

---

*Recall is the proportion of relevant documents retrieved, and precision is the proportion of retrieved items actually relevant. Ideally one would like to retrieve all relevant and reject all nonrelevant to produce recall and precision values equal to 1. When recall is plotted against precision, as in a standard recall-precision graph, curves close to the upper right-hand corner represent superior performance, since both recall and precision are then maximized.

**The writer is indebted to Mr. C. W. Cleverdon for making available the Cranfield collection together with the various relevance assessments.

Relevance Judgments	Generality (Average Number of Relevant per Query)	Percent Overlap with "A" Judgments $\left[\frac{A \cap x}{A \cup x}\right]$
Original Judgments "A"	4.70	100.00%
"B" Judgments	4.28	80.74%
"C" Judgments	11.94	37.09%
"D" Judgments	11.70	37.83%

Relevance Assessments

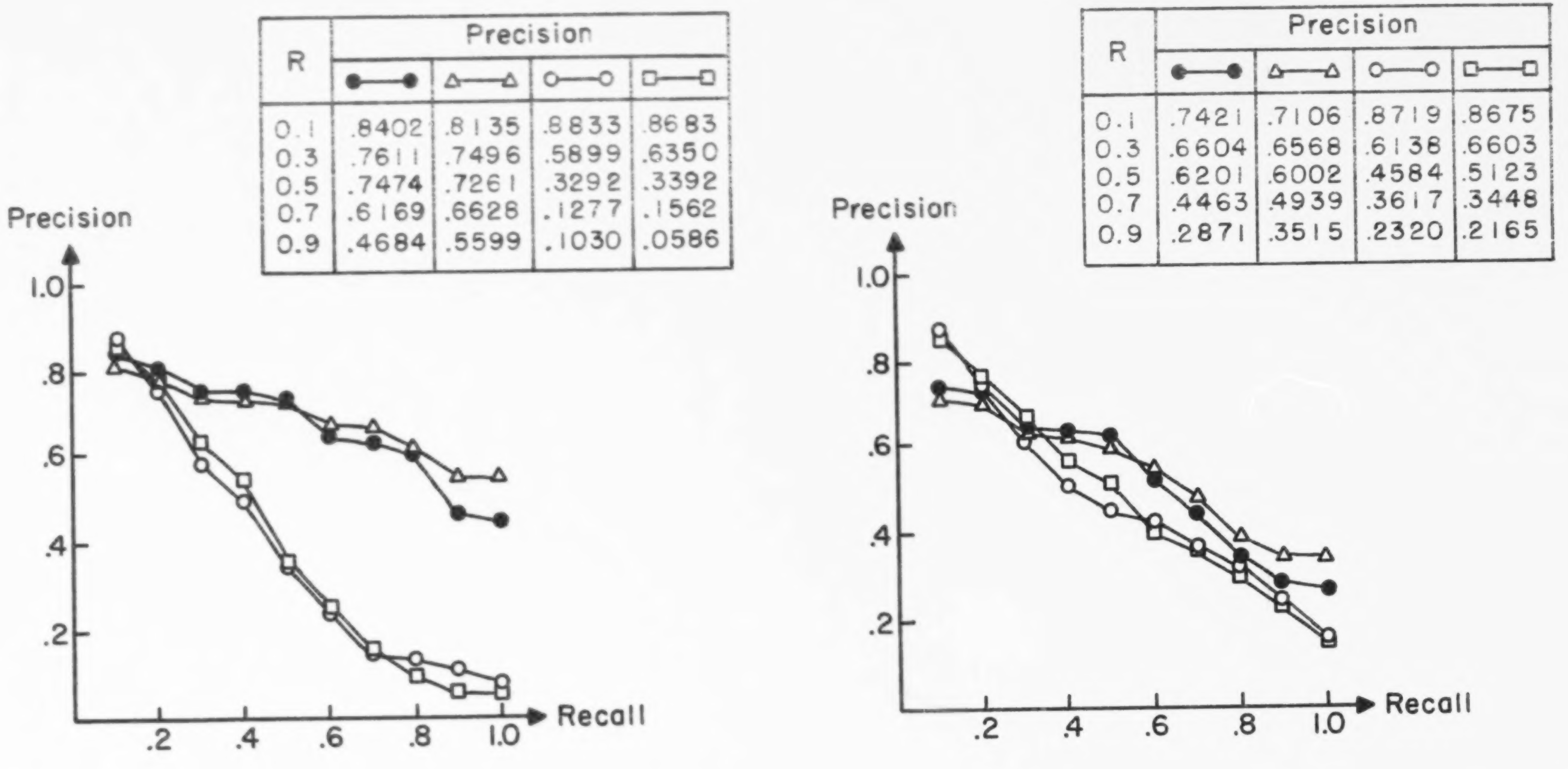
Table 1

to five relevant items per query — corresponding to a strict interpretation of relevance; furthermore the A and B assessments are very similar in nature in view of the overlap of over 80 percent in the respective sets of relevant items per query;

b) sets C and D exhibit much higher generality — almost 12 relevant items per query — corresponding to a less narrow relevance interpretation, and the similarity with the original A judgments is much smaller.

Under normal circumstances, one would expect a better recall-precision performance for the high-generality case, while for equivalent generality, the best relevance assessments would produce the best performance [12]. The actual retrieval effect of the four types of relevance assessments is outlined in the graphs of Figure 3.

It may be seen that when citations only are used in query and document vectors (the 'y' portions), the low generality A and B assessments give much superior performance (Figure 3 (a)). On the other hand, when standard thesaurus concepts are used in addition to citations, as in Figure 3(b), the differences among the four types of assessments largely disappear. The same is true when the thesaurus alone is used for analysis purposes (without the additional citations). The latter results are in agreement with earlier studies showing that only minor differences occur in averaged recall-precision graphs with normal variations in relevance assessments. [13] The large differences in the performance of the "citations only" run of Figure 3(a) must then be due to the peculiar nature of relevance assessments 'A' and 'B', and to the special treatment accorded to the source document citations during the relevance judging procedure. For practical purposes, it appears safer to use the 'C' and 'D' judgments in assessing the relative importance of



R	Precision ●—●	Precision △—△	Precision ○—○	Precision □—□
0.1	.8402	.8135	.8833	.8683
0.3	.7611	.7496	.5899	.6350
0.5	.7474	.7261	.3292	.3392
0.7	.6169	.6628	.1277	.1562
0.9	.4684	.5599	.1030	.0586

a) Citations Only (Citations of Query Source Doc. Used as Query)

R	Precision ●—●	Precision △—△	Precision ○—○	Precision □—□
0.1	.7421	.7106	.8719	.8675
0.3	.6604	.6568	.6138	.6603
0.5	.6201	.6002	.4584	.5123
0.7	.4463	.4939	.3617	.3448
0.9	.2871	.3515	.2320	.2165

b) Thesaurus with Citations

Effect of Relevance Assessments on Citation Indexing
(200 documents, 42 queries)

●—● Original 'A' Relevance Judgments
△—△ 'B' Relevance Judgments
○—○ 'C' Relevance Judgments
□—□ 'D' Relevance Judgments

Figure 3

citation data and standard subject indicators in a retrieval environment.

The main output results are shown in Figure 4 for both 'A' and 'C' relevance assessments. It may be seen that in both cases the augmented thesaurus vectors, obtained by adding citation concepts to standard subject indicators, improve the precision performance by up to ten percent for a given recall point. The short "citations only" vectors provide superior performance for the 'A' relevance assessments for the reasons already stated. Even with the 'C' judgments, the citation indexing alone provides a very high standard of performance in the low recall range.

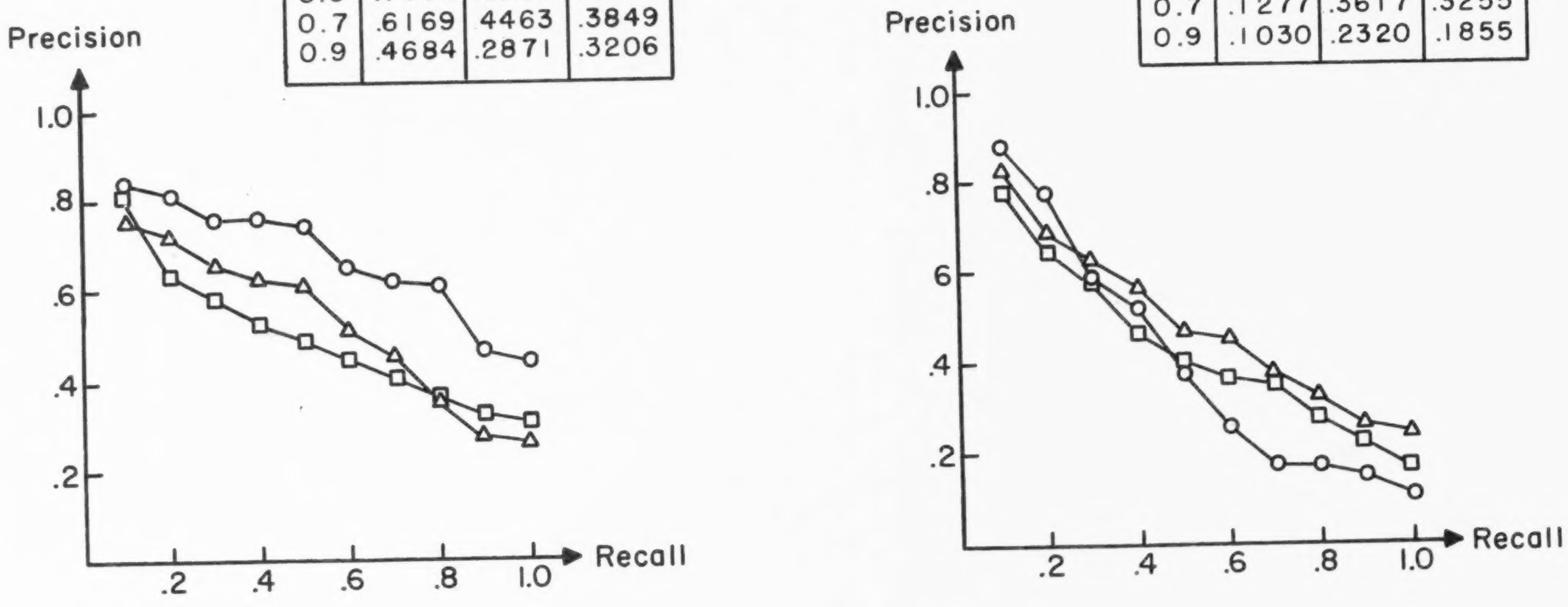
The usefulness of bibliographic citations for content analysis purposes is further illustrated by the output of Figure 5 in which a standard word stem matching process is compared with the word stem vectors augmented by citation information. It can be seen from the output of Figure 5(a) that the augmented stem vectors generally produce better performance then the standard word stems; this confirms the results obtained in Figure 4 for the thesaurus process. Furthermore, the output of Figure 5(b) shows that augmented thesaurus vectors are slightly preferable to augmented word stem vectors.

The performance data of Figures 3 to 5 were obtained by adding source document citations to the normal query formulations. Since the source documents exhibit especially strong relevance characteristics — each user knows in advance that the source documents are immediately germane to the information queries — an attempt was made to relax the requirement for source document citations by replacing them by the citations attached to a randomly chosen relevant document.

Specifically, each query is first processed in the standard manner using a normal thesaurus look-up procedure. A document identified as relevant

R	Precision		
	○—○	△—△	□—□
0.1	.8402	.7421	.8144
0.3	.7611	.6604	.5733
0.5	.7474	.6201	.4811
0.7	.6169	4463	.3849
0.9	.4684	.2871	.3206

R	Precision		
	○—○	△—△	□—□
0.1	.8833	.8719	.7648
0.3	.5899	.6138	.5197
0.5	.3292	.4584	.4064
0.7	.1277	.3617	.3255
0.9	.1030	.2320	.1855



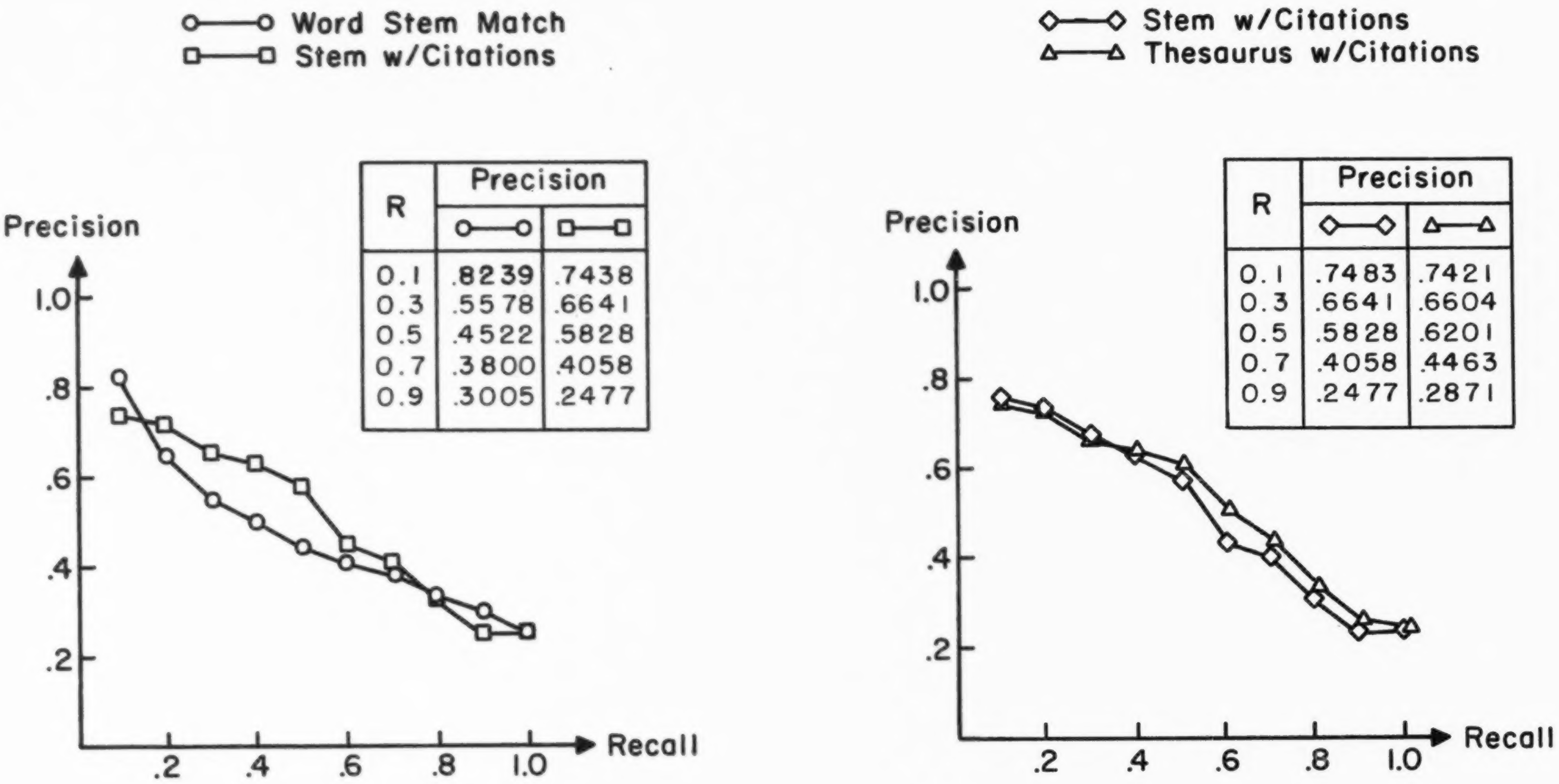
a) 'A' Relevance Judgments

b) 'C' Relevance Judgments

Comparison of Citation Indexing with Thesaurus Operations
(200 documents, 42 queries; source documents)

○—○ citations only
△—△ thesaurus with citations
□—□ thesaurus (Harris 3)

Figure 4



R	Precision	
	○—○	□—□
0.1	.8239	.7438
0.3	.5578	.6641
0.5	.4522	.5828
0.7	.3800	.4058
0.9	.3005	.2477

a) Word Stems with Citations

R	Precision	
	◇—◇	△—△
0.1	.7483	.7421
0.3	.6641	.6604
0.5	.5828	.6201
0.7	.4058	.4463
0.9	.2477	.2871

b) Word Stems vs. Thesaurus

Effect of Citations on Word Stem Matching Process
(200 documents, 42 queries; source documents)

Figure 5

after the fact — but not known to the user in advance — is then used in lieu of the normal source document, and citations from this relevant document are used to form the augmented query vector. The relevant documents chosen for this purpose are eliminated from the document collection for evaluation purposes. The output of Figure 6 shows that the citations obtained from the randomly chosen relevant documents do not have sufficiently strong relevance characteristics to lead to an improved retrieval performance over and above the standard thesaurus method.

The following principal results emerge from the present citation test:

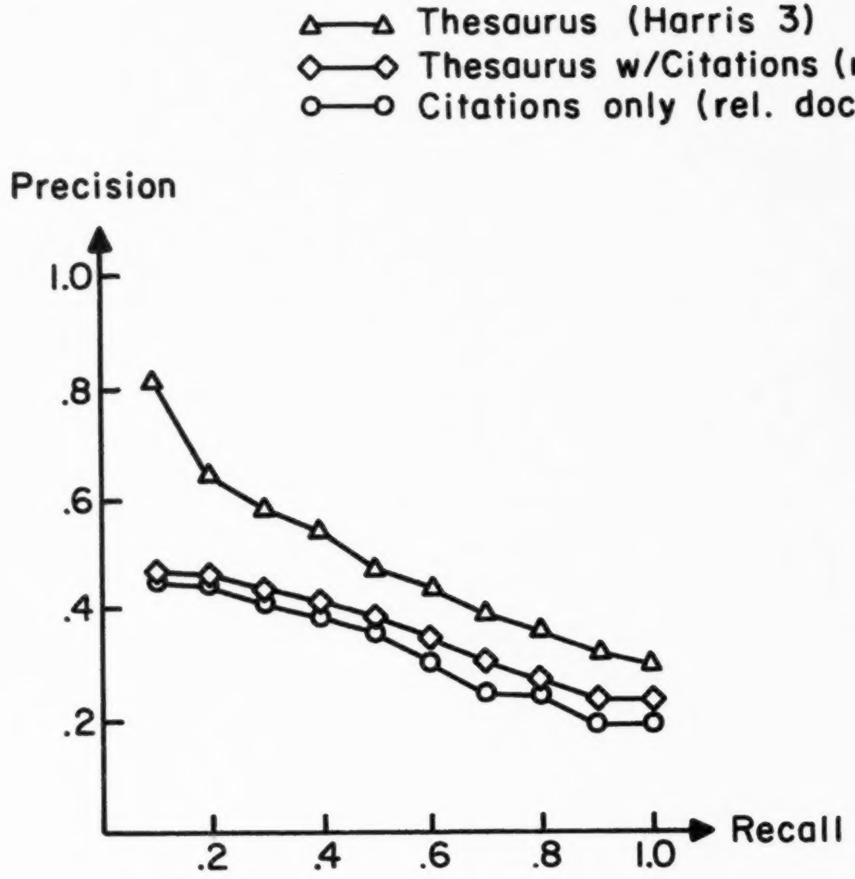
a) the general usefulness of bibliographic citations for document content analysis, previously noted by a number of other investigators, is entirely confirmed;

b) bibliographic citations used for document content identification provide a retrieval effectiveness fully comparable to that obtainable by standard subject indicators at the low recall-high precision end of the performance range;

c) the augmented document vectors, consisting of standard concepts plus bibliographic citation identifiers appear to provide a considerably better retrieval performance than the standard vectors made up of normal subject indicators only;

d) the bibliographic citations attached to information requests should be taken from documents whose strong relevance characteristics to the respective queries is known in advance by the user population.

The present experiment then leads to the conclusion that documents processed in a retrieval system should normally carry bibliographic citation codes in addition to standard content indicators. When queries are received from the user population, improved service can be obtained by using document citations as part of the query formulations whenever documents with



R	Precision		
	△—△	◇—◇	○—○
0.1	.8144	.4680	.4390
0.3	.5733	.4364	.4112
0.5	.4811	.3915	.3615
0.7	.3849	.3084	.2518
0.9	.3206	.2361	.1935

Use of Citations from Random Relevant Documents
(200 documents, 42 queries)

Figure 6

a priori relevance characteristics are identified by the users at the time of query submission. If no documents with strong relevance characteristics are available when the query is first received, bibliographic citations can still be used as a feedback device by updating the query formulations with citations from previously retrieved relevant documents.

References

[1] J. H. Westbrook, Identifying Significant Research, Science, Vol. 132, No. 3435, October 28, 1960, p. 1229-1234.

[2] J. Margolis, Citation Indexing and Evaluation of Scientific Papers, Science, Vol. 155, No. 3767, March 10, 1967, p. 1213-1219.

[3] D. J. de Solla Price, Networks of Scientific Papers, Science, Vol. 149, No. 3683, July 30, 1965, p. 510-515.

[4] E. Garfield, Citation Indexes for Science, Science, Vol. 122, No. 3159, July 15, 1955, p. 108-111.

[5] M. M. Kessler, Comparison of the Results of Bibliographic Coupling and Analytic Subject Indexing, American Documentation, Vol. 16, No. 3, July 1965, p. 223-233.

[6] G. Salton, Associative Document Retrieval Techniques using Bibliographic Information, Journal of the ACM, Vol. 10, No. 4, October 1963, p. 440-457.

[7] J. W. McNeill and C. S. Wetherell, Bibliographic Data as an Aid to Document Retrieval, Scientific Report No. ISR-16 to the National Science Foundation, Section VIII, Cornell University, September 1969.

[8] M. Amreich, G. Grissom, D. Michelson, E. Ide, An Experiment in the Use of Bibliographic Data as a Source of Relevance Feedback in Information Retrieval, Report No. ISR-12 to the National Science Foundation, Section XI, Cornell University, June 1967.

[9] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill Book Company, New York, 1968.

[10] R. G. Crawford and H. Z. Melzer, The Use of Relevant Documents instead of Queries in Relevance Feedback, Scientific Report No. ISR-14 to the National Science Foundation, Section XIII, Cornell University, October 1968.

[11] C. W. Cleverdon and E. M. Keen, Factors Determining the Performance of Indexing Sustems, Vol. 2, Test Results, Aslib Cranfield Research Project, Cranfield, 1966.

[12] G. Salton, The Generality Effect and the Retrieval Evaluation for Large Collections, Scientific Report No. ISR-18 to the National Science Foundation and to the National Library of Medicine, Section II, Cornell University, October 1970.

[13] M. E. Lesk and G. Salton, Relevance Assessments and Retrieval System Evaluation, Information Storage and Retrieval, Vol. 4, No. 4, October 1968, p. 343-359.

Appendix

Sample Citation Codes

I. Sinnott, Colin S., "On the Prediction of Mixed Subsonic/Supersonic Pressure Distributions," Journal of Aerospace Sciences, Vol. 27, p. 767, 1960.

SINOJAS27076760

II. Herriot, John G., Blockage Corrections for 3-Dimensional Flow Closed-Throat Wind Tunnels, with Consideration of the Effect of Compressibility, NACA, Rep. 995, 1950.

HERNACAOR099550

III. Cheng, H. K., "Hypersonic Shock-Layer Theory of the Stagnation at Low Reynolds Number," Proceedings of the 1961 Heat Transfer and Fluid Mechanics Institute, Stanford University Press, Stanford, California, 1961.

CHEPHTFOOHYPE61

IV. Couper, J.E., The Operation and Maintenance of Recorder Type IT 3-16-61, Unpublished M.O.A. Report.

CØUUNPUØPERAT**

V. Goldstein, S., Ed., Modern Developments in Fluid Dynamics, Vol. I, p. 135; Oxford, The Clarendon Press, 1938.

GØLMØDERNDE0138

IV. Automatic Resolution of Ambiguities from Natural Language Text

S. F. Weiss

Abstract

This study investigates automatic disambiguation by template analysis. The evolutionary process by which ambiguities are created is discussed. This leads to a classification of ambiguities into three classes: true, contextual, and syntactic. The class assigned to a given word is dependent on the syntactic and semantic functions performed by the word. Only true ambiguities are suitable for automatic resolution.

In this study, automatic disambiguation is accomplished by an extended version of template analysis. The process consists in locating an ambiguous word and in testing its environment against a predetermined set of rules for occurrences of words and structures which indicate the intended interpretation. Experiments using this process show that a high degree of accuracy in resolution can be achieved.

The process under consideration is not completely automatic because it requires that a set of disambiguation rules be created *a priori*. The creation of this rule set, however, is sufficiently straight forward that it may eventually be done automatically. A learning program is implemented to accomplish this. The process reads input words and attempts to resolve any existing ambiguities. If a resolution of the ambiguity is performed incorrectly, the rule set is augmented and modified appropriately, and the next input is considered.

The experimental results obtained are poor for the first few inputs. The performance steadily improves as more inputs are processed, and finally

levels off at above 90% accuracy. A true learning process is thus indicated.

The proposed learning process is not only useful for disambiguation, but can also serve for a number of other applications, where it may be desired to tailor a process to a particular user need.

1. Introduction

An ambiguous word is defined as a word which can have two or more different meanings. There exist a great many such ambiguous words and their occurrence in text is fairly common. In general they create no problem for a human reader because he is constantly aware of the context of the material he is reading and of the real world. This usually makes obvious the proper definition of an ambiguous word. For example, the word BOARD may mean, among other things, a piece of wood or a group of people. In the first of the two sample sentences below, the ambiguity is resolved by the context of the sentence while in the second, resolution is achieved by the reader's knowledge of the real world. In other words the reader knows from his general knowledge that it is much more likely to cut a piece of wood than a group of people, even though it is technically possible to do both.

A: He is a member of the board of directors.

B: He cut up the board.

Disambiguation by computer is considerably more difficult. A computer does not automatically conceptualize the context of the text as it is read. Also a computer cannot be expected to contain the vast store of knowledge that a human reader possesses. This study presents some techniques for automatic semantic disambiguation of words from natural language text and the application of template analysis to this process. A complete discussion of template analysis is presented in Weiss [16].

The justification for such a study is that ambiguities in text are detrimental to any natural language process which uses that text. The extent of the damage imposed by ambiguities varies with the natural language process as is shown by the three examples below.

1. In a SMART-like information retrieval system ambiguous words are assigned multiple concepts to represent their various possible definitions. Since only one of the definitions is is actually correct, this process adds erroneous material to the document and query vectors. But this is not a serious problem since ambiguous concepts are rare and thus make up only a small part of a document or query vector. Resolution of ambiguities makes a very small change in a concept vector and hence causes only a very small change in document-query correlations. Thus in a retrieval environment, ambiguities may not pose a very serious problem and are hardly worth resolving. Examples 2 and 3 present environments in which the consequences of ambiguities are more serious and disambiguation is more justified.

2. A serious problem in automatic syntactic analysis is that an analyzer may produce many analyses for a single input. It is very difficult if not impossible to determine the intended analysis from among this set. Thus syntactic analysis schemes which generate as few analyses as possible are clearly the most desirable. One cause of multiple analyses is words which have more than one syntactic role. For example, the word FLYING can be either a verb or an adjective. This in turn gives rise to several analyses of

   THEY ARE FLYING PLANES.

   Some systems perform semantic tests to determine which of the syntactic analyses is semantically feasible. An even better approach is to resolve ambiguities prior to syntactic analysis thus reducing the number of analyses produced. It sometimes happens that syntactically ambiguous words are also semantically ambiguous. NEGATIVE for example is usually an adjective when it means NOT and a noun in the photographic context. Thus by resolving the semantic ambiguity, the syntactic ambiguity is also removed. In this way resolution of semantic ambiguity can reduce the number of analyses resultant from an automatic syntactic analysis scheme and hence simplify the task of determining the correct analysis.

3. In natural language command analysis or a natural language programming language, each statement must be mapped into a unique command or command sequence. Statements which due to ambiguities simultaneously specify more than one command sequence are unexecutable. Current programming languages such as FORTRAN and ALGOL deal with this problem simply by prohibiting all but the most trivially resolvable ambiguities (such as the minus sign which may be unary or binary). This is not possible in natural language command analysis and thus all ambiguities must be resolved before execution is possible.

These three examples show how the problems caused by ambiguities in natural language text vary according to the application. In the third example resolution is a necessity while it is more or less a convenience in the other two. In general it appears that at best, ambiguities do no harm and at worst they are disasterous. In no case do they ever seem to have constructive effects. Of course there are other examples of consequences of ambiguities but these three seem sufficient to justify further investigation into the area of automatic disambiguation.

2. The Nature of Ambiguities

Most words in isolation do not have a well defined meaning. The exact meaning of a word is formed by the interaction of the word and its context. Each word is both acted upon by its context and acts upon its context. The action that a word performs on its context is called its semantic function. This can be thought of as a mathematical function with the word's context as its argument and the total meaning as its value. An example is presented in Figure 1 below.

Phrase: Bottom of the bottle
Word: Bottom
Semantic function: indicates lowest point in context
Context: "of the bottle"

Application of semantic function to context yields the value: lowest point in the bottle

Example of Semantic Function

Figure 1

Building on the concept of semantic function it is now possible to define three types of ambiguities. A word is a true ambiguity if it has two or more distinct semantic functions. An example is the word DEGREE. This may refer to a unit of temperature or angle as in "a 90 degree turn" or an award from a school as in "college degree". These are clearly two separate semantic functions. Some words have only one semantic function yet still appear ambiguous. This situation is produced when a single semantic function, acting on a variety of contexts, produces vastly different meanings. Such words are termed contextually ambiguous. As an example, the word CORE is considered ambiguous in the ADI dictionary. It refers to both a computer memory and the central part of something. However there is only one semantic function at work here and it designates the central aspect of its context. A computer memory is at least conceptually if not physically the center of a computer. Thus CORE is a contextual ambiguity according to the definition above.

A third type of ambiguity is syntactic ambiguity. The meaning of such a word is dependent upon its syntactic role. The meaning of ELABORATE, for example, differs somewhat depending on whether it is used as a verb or adjective. These differences in meaning, however, are generally just slight variations of a single semantic concept.

The classification of an ambiguous word into one of these categories is not a strictly defined process. The categories are not completely disjoint; and the ambiguous words themselves are in a constant state of evolutionary change much like biological evolution. A good example of the development of an ambiguous word can be seen in the word BOARD which can mean a piece of wood, a group of people (board of directors), or food (room and board). Originally board referred only to a piece of wood or a table. Because of their close relation to the table, the people who met there and the food served on it became associated with the board. In time this connection disappeared and BOARD currently appears to have three separate meanings. In general, ambiguities seem to stem from idioms and associations due to similarities such as between the food and the table on which it is served. These words gradually evolve into contextual and finally true ambiguities. Many of the words currently considered contextually ambiguous may eventually become true ambiguities. For example, it is conceivable that in the future, computer memories may no longer be considered a central element of the machine Thus CORE, shown previously to be a contextual ambiguity, may become a true ambiguity. As another example, consider the word LUNACY. It was originally thought that this form of insanity was caused by the moon and hence the name. Now, however, the lunar influence is better understood, and there is no connection between the disease and the moon. Thus the common stem LUNA represents an evolved ambiguity.

Before considering resolution of ambiguities, it is necessary to decide which type or types can and should be resolved. There are several criteria for this decision. First, does the resolution of the ambiguity add any additional information to that already known? Second, does the added information warrant the work involved to determine it? And finally, what harmful effects might be expected if the ambiguity were not resolved?

As shown above the meanings of the various forms of a syntactic ambiguity vary only slightly. Thus very little information is added if resolution is performed. Also, harmful effects caused by syntactic ambiguities are slight and occur only in special cases as is shown in the following example. Let A, B, and C be words with A syntactically ambiguous and having meanings in thesaurus classes 1 and 2 (see Figure 2). B and C are not ambiguous. B is in thesaurus category 1 and C is in 2. Leaving A unresolved, that is using only a single concept to represent A, would in effect combine categories 1 and 2. This would make B appear synonymous to C which is not really the case. However, as shown previously, the differences in meaning of the various forms of syntactic ambiguities are slight thereby necessitating categories 1 and 2 being very close in meaning. Thus combining B and C is not a particularly grave error. For this reason it appears unwarranted to resolve syntactic ambiguities.

WORDS	THES. CATEGORIES
A	1,2 (SYN AMB)
B	1
C	2

Sample Syntactic Ambiguity
Figure 2

As discussed previously, contextual ambiguities have only one semantic function. The differences in meaning are caused by the context rather than by the word itself. It is therefore questionable whether such words should be disambiguated at all. Also because contextual ambiguities derive much of their meaning from context, they may have a broad spectrum of meanings rather than the few discrete meanings possessed by most true ambiguities. Intuitively at least this seems to indicate that the resolution of contextual ambiguities is both more difficult and less precise than resolution of true ambiguities. Experiments in this area show this to be the case.

The remaining class, the true ambiguities, demonstrates the properties necessary to justify their resolution. The remainder of this study deals with techniques for automatic resolution of true ambiguities.

3. Approaches to Disambiguation

Many automatic natural language analysis systems have a facility for automatic disambiguation. For some this entails the use of semantic information to resolve syntactic ambiguities and hence reduce the number of syntactic parses. Other systems actually tackle the problem of true semantic ambiguities. This section discusses some of these approaches to automatic disambiguation.

The easiest solution to the problem is simply to ignore it. This approach is actually not as absurd as it initially appears. When the domain of discourse is sufficiently limited, many ambiguities disappear. This is the case with the information retrieval system implemented by Dimsdale and Lamson [3]. By limiting the subject area to the medical field, the problem of ambiguities solves itself. For example, the word CELL has a number of possible meanings (dry cell, jail cell, muscle cell). However, only one of these interpretations is appropriate to medicine; and thus in this context, CELL may be treated as an unambiguous word.

As mentioned previously, one possible application for automatic disambiguation is in indexing documents for information retrieval. There are a number of possible techniques. Some researchers, for example Ranganathan [10] and Mandersloot et. al. [4], suggest that ambiguous words be represented by a number of concepts which resolve the ambiguity. One of these additional concepts could be the hierarchical father of the word under consideration. For example, the ambiguity caused by the word TYPE could be resolved by adding the concept for PRINTING. SMART uses a different method. An ambiguous word is assigned the concepts of all its possible interpretations. The set of concepts then share the total weight. Thus SMART covers all possibilities and is guaranteed of having the correct concept. However it is also guaranteed of having some wrong concepts. This inclusion of error would appear to weaken the indexing scheme and hence damage retrieval; but this is not the case. The occurrence of ambiguous words is quite rare and hence the error introduced by the process represents only a very small part of a total concept vector. Thus the effect on results is very small. In addition problems can only be caused when a thesaurus is used that contains words which are synonymous to some but not all of the interpretations of an ambiguous word. Actual experiments reveal that the resolution of ambiguities in SMART concept vectors results in improvement of less than 1%. Thus the added effort required to resolve ambiguities in this type of information retrieval context seems unwarranted.

Some question-answering systems with a restricted data base are able to disambiguate simply by testing the various interpretations against the data base and choosing the one that is applicable. DEACON is an example of one such system [15]. A query such as the one below is ambiguous since Guam is an island and an aircraft carrier. But since DEACON's data base deals with ships, the latter interpretation is chosen.

How many people are on Guam?

Other systems perform a similar type of disambiguation by using lists of true predicates. Coles' system, for example, tests the query against a set of truth values. Similarly the process used by Schank and Tesler tests various ambiguous interpretations for consistency with a set of real world attributes.

Another basic method for automatic disambiguation is to associate semantic features with each word in the lexicon. Rules, similar to syntactic rules, can then test various possible interpretations for semantic as well as syntactic wellformedness. One such system is Simmons' PROTOSYNTHEX [12]. Each word is associated with its semantic class. For example, "angry" is a type of emotion and "pitcher" is a type of person (baseball player) or a type of container. Ambiguities such as "pitcher" are resolved by testing its syntactic structure against a set of semantic event forms. These indicate possible valid relationships between semantic classes. The semantic event forms reveal, for example, that a person can have an emotion while a container cannot. Thus the disambiguation of "angry pitcher" is accomplished. Woods accomplishes disambiguation in much the same way. Syntactic and semantic features are attached to words; and rules indicate legitimate combinations of these features.

Lesk uses a similar approach in his proposed natural language analysis system, but with a unique statistical feature [7]. In his system words are assigned both syntactic and semantic role indicators by the dictionary. The parse then determines syntactic dependencies and tests them for semantic validity. Those interpretations which fail the semantic test are eliminated thus accomplishing some disambiguation. In addition, each interpretation of each ambiguous word has associated with it the probability of the "correctness" of that interpretation. For example, in a sports text the word "base" would be much more likely to refer to a baseball base than to a military base; and probabilities may be assigned accordingly. During the syntactic analysis a number of possible parses are developed. The probability of correctness for each is the product of its constituent probabilities. In this way, interpretations with very low probabilities of being valid may be eliminated thus accomplishing another form of disambiguation.

The processes presented above use syntactic and semantic features to qualify the words and then employ a common rule list to govern word combination. A more detailed approach to disambiguation is to attach specific combination rules to each word. The need for this can be seen in the following simple example. Most noun phrases consisting of an adjective and a noun assume the basic features of the noun. The phrase may then be used anywhere that the noun is legal. For example, the phrase "folding money" may be used wherever "money" can be used. This is not true for "folding" which in some sense loses its identity when combined with the noun. Most of the systems which use a combination rule list can determine this property. There are, however, exceptions to this rule. Consider the phrase "Tompkins County". Here the word "Tompkins", acting as an adjective, dominates the phrase. It is all right to say "Buffalo is in a county" but "Buffalo is in Tompkins County" is semantically and geographically incorrect. Thus in this case the phrase assumes the properties of the adjective. To treat properly this and other similar cases, it is useful to associate combination rules with individual words rather than using a common rule list for all words. Some of the automatic systems which employ this approach are those by Kellogg [6], and Quillian [9].

Kellogg's scheme assigns a set of data structures to each interpretation of each word. These include semantic features and selection restrictions. For a particular word the selection restrictions limit the words with which it can be associated to only those with specific semantic features. For example, the verb "talk" can take only an animate subject.

In Quillian's Teachable Language Comprehender, memory is represented as an interconnected network of nodes. The meaning of a phrase is determined by locating a path in the network from one constituent word to the other. For some phrases there are more than one legal path. This indicates an ambiguous phrase. Disambiguation is achieved by using the shortest path. This represents the most likely interpretation and is thus similar in approach to Lesk's statistical scheme.

The processes discussed so far deal with disambiguation as a tool in some sort of information retrieval or question-answering facility. Moyne [8] summarizes this type of disambiguation as falling into one of four interaction types: interaction with the lexicon, with the data base, with the general system capabilities, and if all else fails, interaction with the user. This last type is strictly a last resort measure but is very helpful when unresolvable ambiguities are encountered.

As shown above, much of the work in disambiguation deals with larger information retrieval and question-answering systems. But some work has also been done on ambiguities alone. In particular is the work by Stone [14], Coyaud [2], and Borillo and Virbel [1]. All these schemes are based on resolution of ambiguities by examination of semantic context. Associated with each word is a set of words and concepts which, if found near the ambiguous word, specify a particular interpretation. Stone concentrates on the resolution of ambiguities in high frequency words such as "matter". The study by Borillo and Virbel represents the most detailed and complete discussion of disambiguation encountered in the literature. They discuss all forms of ambiguities, and present for each, the methods needed for resolution. Ambiguous words are divided into five classes:

1. key word
2. grammatical ambiguity
3. semantic ambiguity
4. combined semantic and grammatical
5. forced

The key words are words of variable importance whose resolution is not vital. The forced words are so important that all interpretations must be represented. The remainder are self explanatory. The third and fourth classes are most interesting and correspond roughly to the true ambiguities presented in the previous section. Resolution is achieved by examing some environment of the ambiguous word for certain structural or semantic clues. In addition, Borillo and Virbel give a suggested list of attributes for a disambiguation process. These are first, that the context of an ambiguous word should be scanned in closest to farthest order. Second, resolution rules should be weighted according to their probability of correctness. And third, the scope of the context should be variable from word to word.

Building on this introduction, the next sections present an automatic disambiguation scheme using the template analysis process. It is designed as a disambiguation package for a natural language conversational system and hence expected input is clearly restricted. In addition, each ambiguous word is treated separately and the relevant context of each word is quite limited. Thus the input seems applicable to template analysis.

4. Automatic Disambiguation

A) Application of Extended Template Analysis to Disambiguation.

Associated with each ambiguous word is a set of keywords or structures which identify the intended meaning. For example, if within the context of the word BOARD, there are references to "fir", "pine", or "oak", a wooden board is probably intended. If "chairman" or "meeting" occurs, board would be taken to mean a group of people. This key to the intended meaning of am ambiguous word is usually found in the immediate context of that word, often in the same sentence. The actual optimal scope of context varies from word to word. Borillo and Virbel indicate that in general, best results are obtained using large sentence groups (document abstracts). In some cases, however, this is too broad and permits erroneous resolution by matching the wrong key. For this reason the scope of context is defined here to be the sentence containing the ambiguous word. Each sentence containing an ambiguous word is scanned for a resolution key. This resolution key may be a word, group of words, or structure, which reveals the intended meaning. The process is implemented using an extended version of template analysis [16]. This section discusses the extensions to template analysis that are required to facilitate automatic disambiguation. The disambiguation process is presented in subsection B and the experimental results in subsection C.

A template is basically a string of words. It matches a natural language input only if a substring of the input matches the template elements exactly including ordering and contiguity. Many ambiguities may be resolved using templates; but for others, templates are too strict a criterion. For these words the presence of a resolution key anywhere in the input is sufficient to warrant resolution. For this reason the context rule is used. Like a template, the context rule is a string of words. However a context rule is considered to match an input if the input contains all the words of the rule with no restriction on ordering or contiguity. In Figure 3 below, the template matches only input A while the context rule matches A, B, and C. Thus a context rule represents a purely semantic test while a template requires both semantics and syntax (structure).

The process used for matching the input against both templates and context rules is a middle-outward search strategy. That is, the search begins at the ambiguous word and extends outward in both directions. This guarantees finding the resolution key which lies closest to the ambiguous word. This is necessitated for two reasons. First, if an input contains two or more occurences of a particular ambiguous word, each must be paired with its closest resolution key in order to obtain correct results. The examples in Figure 4, though admittedly rather contrived, demonstrate the need for this technique.

B) The Disambiguation Process

The process of disambiguation requires the following elements: a small thesaurus of words needed in the disambiguation process, a set of templates and a set of context rules. The process first reads an input and each word is looked up in the disambiguation thesaurus. Most words are not found and are classified as unknown. The input is first matched against the template set and then the context rule set using the middle-out search strategy. Disambiguation is performed by the first rule successfully matched. The rules in each set are ordered so that the strongest rules, that is the ones that are expected to provide the best disambiguation performance, appear at the top. The weaker or last resort rules appear near the bottom. Scanning the rule list top to bottom matches strong rules before weak rules. This critical ordering essentially weights the rules and

Template: COMPUTER PROGRAMMING

Context rule: COMPUTER PROGRAMMING

Inputs:
A. Elements of computer programming
B. Programming of digital computers
C. Computer design and programming

Template matches A only

Context rule matches A, B, and C

Comparison of Templates and Context Rules

Figure 3

Input A.	It was very cold when he received his college degree.
Action:	COLLEGE rather than the temperature reference must be used to disambiguate DEGREE.
Input B.	His college degree was to a large degree, well earned.
Action:	Each DEGREE must be associated with its nearest resolution key.

Search Strategy (Underline Indicates Resolution Key)

Figure 4

ensures that an input is matched with the rule that has the greatest chance of providing a correct analysis. Associated with each rule is the meaning appropriate to that resolution key. If no match is found between an input and any rule, the ambiguity is considered unresolved. An option may be used in connection with such unresolved inputs. For some ambiguous words one interpretation is much more likely than all the rest. For these a significant saving in the size of the rule sets and in the work involved can be obtained by testing for all but the most likely interpretation. If no matches occur the result is taken by default to be the most likely meaning. This option is used for some of the experiments that follow.

C) Experiments

After classifying the ambiguous words found in the ADI[1] dictionary as true, contextual or syntactic, five true ambiguities are chosen for experimentation. The words are:

DEGREE

TYPE

VOLUME

BOARD

CHARGE

For each word except DEGREE a corpus of 50 sentences is used. A larger corpus is used for DEGREE to provide a more exhaustive test. Each corpus contains all sentences from the ADI documents which contain the ambiguous word as well as other sentences written by the author and other informants. Each corpus is divided into two sets: S-1, called the creation set, and S-2, called the test set. S-1 contains 20 sentences, S-2 contains the remainder of the corpus. The experimental procedure used for each word is as follows. First, using S-1 only, a thesaurus, template set and context rule set are created by hand. The disambiguation program is then run on S-1. Appropriate additions and modifications are made to the thesaurus and rule sets, and the program is tried again. This continues until the process provides a high degree of success in resolving ambiguities from S-1. The thesaurus and rule sets existing at this point are thus effectively tuned to the creation set S-1. Next, and without further modification of the thesaurus or rules, the disambiguation process is run using S-2 as input. The process is thus tested on an input set it has never seen before, and one to which it is not specifically tuned. The result parameters used are shown in Figure 5 below. Resolution recall indicates what proportion of the total number of ambiguities in the input set are correctly resolved, while resolution precision indicates what proportion of the analyses performed by the system are correct. In order to perform satisfactorily, the process must give reasonably high values for both RR and RP. In the optimal case both values are 1. The results obtained for the five S-2 sets appear in Figure 6 along with totals for all five words. The default option is used in the analysis of TYPE and CHARGE. Inputs for which the system does not perform an analysis for these words are taken to be of a particular interpretation. Thus no inputs are considered unanalyzed (indicated in Figure 6 by an asterisk in the U column).

---

[1] The ADI Collection is a set of short papers on automation and scientific communication published by the American Documentation Institute, 1963.

These results indicate that extended template analysis is a useful and accurate technique for resolution of true ambiguities. The errors which do occur are not, in general, generated by inputs with normal constructions. Rather they are due mostly to idiomatic expressions which are not included in

T	The Total number of ambiguities in the input set. (This number is sometimes larger than the number of sentences in the input set because a few of the sentences contain multiple occurrences of the ambiguous word).
C	The number of ambiguities correctly resolved
I	The number of ambiguities incorrectly resolved.
U	The number of ambiguities not resolved in any way.
RR	Resolution Recall = C/T
RP	Resolution precision = C/(C+I)

Result Parameters

Figure 5

WORD	T	C	I	U	RR	RP
DEGREE	92	84	4	4	.92	.93
TYPE	30	29	1	*	.97	.97
VOLUME	30	27	1	2	.90	.96
BOARD	30	22	0	8	.73	1.00
CHARGE	32	30	2	*	.94	.94
TOTAL	214	192	8	14	.90	.96

* indicates default used

Results of Disambiguation of S-2 Sets

Figure 6

in the creation set. As an example the expression ON BOARD is not in S-1 for BOARD. This in turn leads to a number of inputs in the test set being unanalyzed. While such idioms in natural language may prevent perfect disambiguation quality, they occur relatively infrequently in practice and thus reduce the system performance only slightly.

D) Further Disambiguation Processes

A number of further processes are suggested by the experiments performed here. First, a statistical weighting can be attached to each resolution. This would represent the probability of correctness of the given rule. The context of the ambiguous word could then be searched for all, not just one, resolution key. For each key found, a correlation is calculated which takes into account the probability of the rule being correct as well as the key's distance from the ambiguous word. The rule with the highest correlation is then used. In this way a strong resolution key can take precedence over a weak one lying closer to the ambiguous word.

A second addition is the use of a variable context. All methods for disambiguation presented here including those by Borillo and Virbel and template analysis use a fixed context size for all words. However the optimal context size varies from word to word. It would thus be better to associate with each word, the context width that works. A third possible future technique is to use antirules. These are rules which if matched, tell what interpretation of the ambiguous words cannot be used. For example, if Y appears in an input, interpretation X is prohibited even if indicators for X are present. These extensions, however, are beyond both the scope and the and the spirit of the present study.

5. Learning to Disambiguate Automatically

A) Introduction

The processes of creating and modifying the sets of templates and context rules as presented in section 4 are relatively straightforward and algorithmic in nature. Rules are constructed from creation set inputs by fairly specific means. Likewise, in rule modification an erroneous rule is removed and replaced by one or more rules which perform correctly. It seems possible that these tasks can be handled by computer. Thus instead of telling the program what to do by manually supplying rules, the system would learn to disambiguate by creating and modifying its own rule sets. The advantages of such a system over one of the type described previously are obvious. First, it eliminates the need for a human analyst to study sample inputs and create template and context rule sets. Second, the system is not static. By learning from inputs and its own mistakes it is constantly improving its performance. This process can even be used to tailor a system to an individual user. Disambiguation rules, or rules for any number of other processes, that are designed by or for a particular user are not always well suited for others. By allowing the system to learn separately from each individual, the particular needs of each user are satisfied. This section discusses some techniques for automatically learning to disambiguate.

B) Dictionary and Corpus

When disambiguation rules are prepared by hand, the words which are to be used in the disambiguation are known in advance. The disambiguation dictionary need only contain these relevant words and thus is quite small. In the learning process, there is no prior knowledge of the words that are to be used to facilitate disambiguation. For this reason a full dictionary containing all the words in the input must be employed initially. This large dictionary, however, is needed only temporarily. After the initial instability of the learning process has settled down and relatively fixed rule sets remain, the words in these rule sets may be used to construct a small disambiguation dictionary which can be used thereafter.

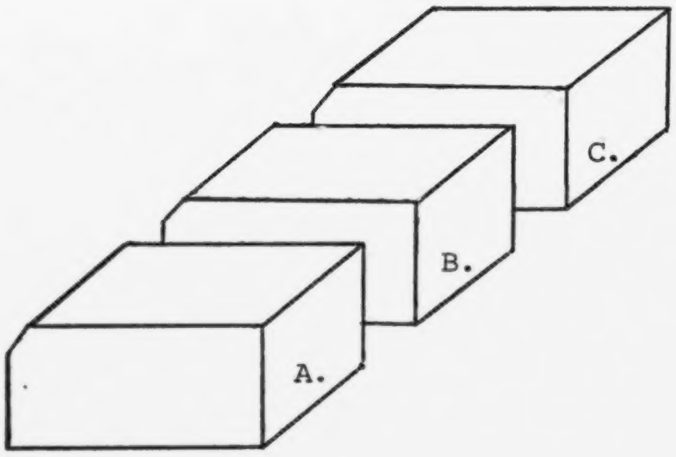
The corpora used in this study are very special. In practice an operational learning system has a very large input set. The learning process may thus extend over hundreds or even thousands of inputs. However, such large data sets are neither readily available nor practical for an experimental system. For this reason it is necessary to develop a small corpus which simulates a much larger one. This is a technique used in a number of experimental studies including Harris' investigation of morpheme boundaries [5]. The rules governing this stem from two fundamental maxims of education. First, a student or learning device cannot be expected to answer a question about something he has not seen previously. That is, a student's first exposure to a concept must be in a learning not a testing environment. And second, to evaluate learning quality, testing is required. Basically these rules say that to test properly a learning system, each concept to be learned must occur at least twice in the input, once for learning and subsequently for testing. Single occurrences are undesirable because if they are considered as a test, they violate the first rule, while if, as the first rule stipulates, the single occurrence is considered for learning only, no testing can occur and the second rule is violated. Large data collections are likely to have multiple occurrences of most concepts. This however is not true for small corpora; and care must be taken to ensure such repetition. To accomplish this the following algorithm is used for corpus construction for each ambiguous word. First, a set of 20 short sentences is written, each containing the ambiguous word. No restriction on vocabulary or construction is imposed for these first 20 sentences. Next, 40 more sentences are written using only words found in the first 20. Again no restriction on construction is imposed. The resulting 60 sentences are sufficiently restricted in vocabulary to ensure that most words and constructs occur at least twice. The corpus thus simulates a corner of a much larger collection. To determine if the system is unlearning previously learned information while learning new material, the actual input consists of the set of 60 sentences repeated three times. Each set of 60 is randomly permuted to eliminate any prejudice due to ordering. The input format is summarized in Figure 7. Such corpora currently exist for three ambiguous words:

DEGREE
TYPE
VOLUME

These are chosen from the set used in previous experiments because VOLUME is rather difficult to disambiguate, TYPE is fairly easy, and DEGREE is between, tending toward difficulty. It is felt that the results obtained and the problems encountered with these words are typical of those to be expected for most other words.

C) The Learning Process

The learning process is implemented as a set of subroutines to the system described in section 4. Dynamic template and context rule lists replace the fixed sets. Initially there are no rules in these sets. The processing of each input sentence proceeds as follows. After the input is



A: Corpus, permutation 1

B: Corpus, permutation 2

C: Corpus, permutation 3

Summary of Input Format for Learning System

Figure 7

read and the ambiguity located, the system attempts to disambiguate the word using templates and context rules currently in the system. When the analysis is complete, the system looks at the correct answer. If the analysis is correct, the system is assumed to contain the appropriate rules for the recognition of the input structure and the system goes on to the next input. If the system is unable to resolve the ambiguity, that is, if no existing rule matches the input, new rules must be added. New templates and context rules are created using the prespecified parameters I and J. I specifies the size of the area around the ambiguous word from which templates are to be made. Similarly J indicates the size of the area from which context rules are to be made. In general J is larger than I since unstructured resolution keys can lie farther away from the ambiguous word than do structured keys. For this study I and J have the values of 2 and 5 respectively. A template is made for each word of the input sentence which lies within plus or minus I of the ambiguous word. The templates preserve the ordering and the relative distance between words. A context rule is created for each word within plus or minus J of the ambiguous word provided the word is not found on a predefined exclusion list. As indicated previously, context rules have no ordering or contiguity restriction. The exclusion list contains words which are of no value in establishing context. These include articles, some prepositions, forms of the verb TO BE, etc. The list is created by consideration of context in general and without any reference to specific words being disambiguated. The exclusion list is not used in the creation of templates because some apparently trivial words are actually important when found in particular structural relationships to an ambiguous word. For example, one of the primary templates for the disambiguation of TYPE is

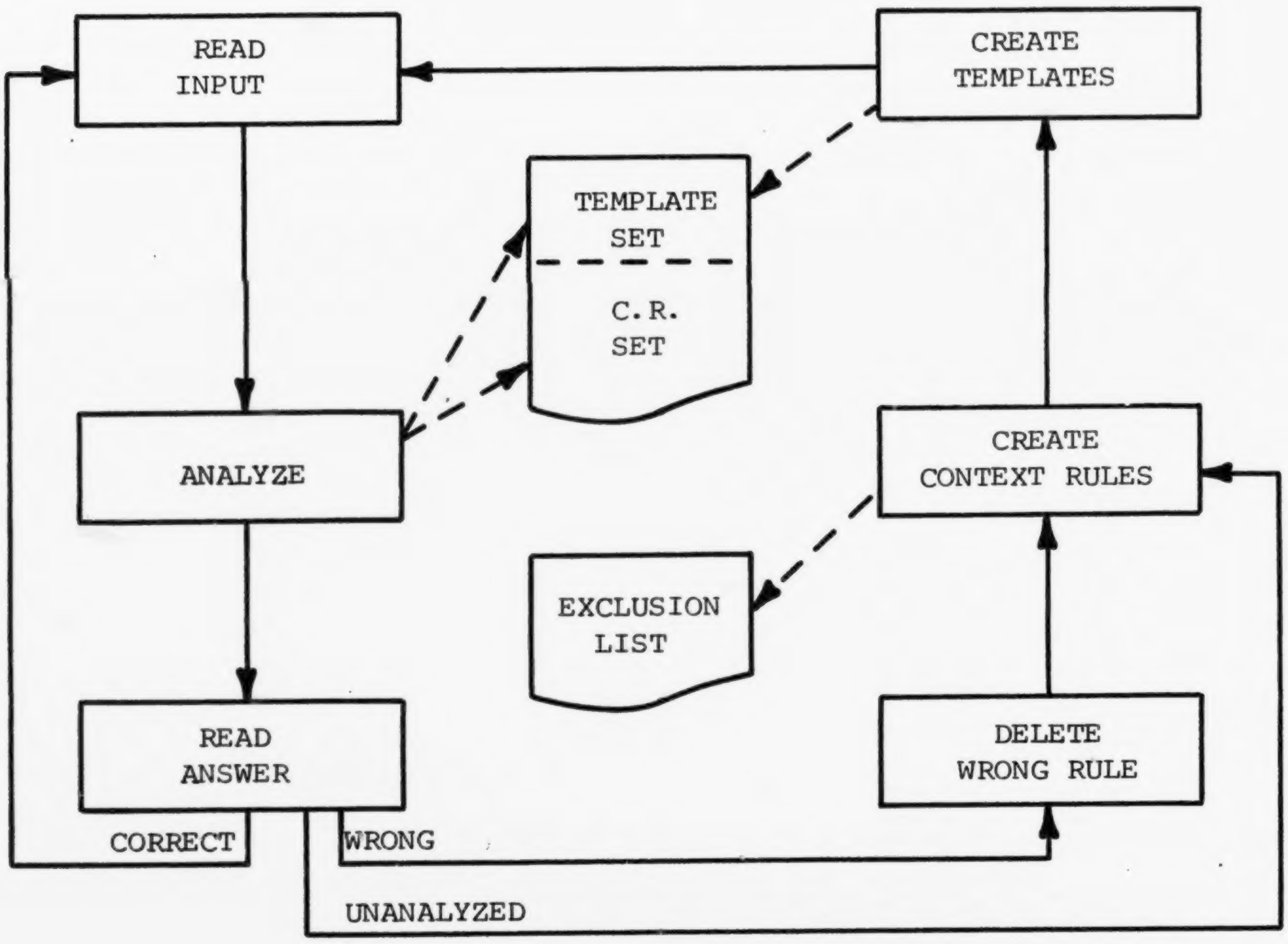
TYPE OF

The templates and context rules created by this process are first placed in a temporary store and checked against rules already in the permanent template and context rule sets. All rules in the temporary store which are not duplicates of existing rules are added to the bottom of the appropriate permanent set. This completes the action for an unanalyzed input.

The third possible outcome is for the system to produce an erroneous analysis. In this case the rule sets not only lack the rules needed for correct analysis, but also contain an erroneous rule. Therefore when this situation arises, the rule which produces the incorrect result must first be removed from the rule set. Each rule lying below the deleted rule is then popped up one position in the rule list. Next, templates and context rules are added just as in the previous case. The operation is summarized in Figure 8.

Critical ordering of rules, as is done in section 4 is not possible when rules are created automatically. However the process of deleting a rule and popping up those below it and then adding the new rules at the bottom tends to make the better rules, that is those which do not get deleted, filter to the top. While this method may not be as effective as critical ordering by hand, it does tend to concentrate the better rules near the top of the lists. The top down search strategy thus matches rules against an input in roughly best first order. Experimental results which verify this are presented later.

Ideally, a system such as that described above operating on a corpus of the form shown in Figure 7 should generate the following type of results. The first few inputs are of course unanalyzed due to the lack of information. As more inputs are read, the overall system performance should begin a steady improvement. Eventually the system should stabilize with a fixed rule set


READ INPUT
CREATE TEMPLATES
TEMPLATE SET
C.R. SET
ANALYZE
CREATE CONTEXT RULES
EXCLUSION LIST
READ ANSWER
DELETE WRONG RULE
CORRECT
WRONG
UNANALYZED

Summary of Learning Process

Figure 8

and near perfect disambiguation. From this point the system should never unlearn. That is, it should never err on an input that it previously analyzed correctly. Likewise it should not be overly sensitive to the order in which inputs are introduced. Actual experimental results obtained compare quite favorably with this idealized behavior. These results are presented in subsection E.

D) Spurious Rules

The learning process presented in part B has a few inherent problems. These center mainly around the treatment of spurious rules. A spurious rule is defined to be a template or context rule which does not discriminate between interpretations of an ambiguous word. As an example, assume that templates and context rules for disambiguation of TYPE are made from input A in Figure 9. One of the templates extracted from this input is LARGE TYPE. This however is of no value as can be seen from input B. Thus LARGE TYPE is considered a spurious rule.

Input A: The book is printed in large type.
(interpretation 1, "printing")

Input B: A tiger is a large type of cat.
(interpretation 2, "kind or variety")

Example of a Spurious Rule

Figure 9

The difficulty with the process as presented in subsection C (to be called version 1 in the remainder of this study), can be visualized by the following example. Assume rules are learned from input A in Figure 9. Included among these is the spurious rule LARGE TYPE which is associated with interpretation 1. Assume also that input B is then processed by a match with LARGE TYPE and hence incorrectly associated with interpretation 1. Version 1 then deletes the interpretation 1 template and substitutes one which is identical except for its association with interpretation 2. Thus a spurious rule is deleted but replaced with one equally spurious. This actually produces a slight improvement since the new rule is inserted at the bottom of the list and thus is less likely to be matched than the one it replaces. But the spurious rules remain and can cause further errors. They may even cause a thrashing back and forth between interpretations and thus prevent stability.

One possible solution to this is implemented in version 2. Whenever a rule is to be deleted because it causes an incorrect analysis, the set of new incoming rules is checked for an occurrence of this same rule. If found, the matched rule is not added to the permanent rule set. Thus using version 2, the incorrect analysis of input B would not only remove LARGE TYPE from the template set but would also prevent this same template (with a different interpretation) from entering the set at that time. In the short run this has the effect of eliminating spurious rules from the system. But since no record is kept, these same spurious rules may reenter the system the next time they occur. A reoccurrence of input A follwoing input B for example, would put LARGE TYPE back on the rule list. Thus while version 2 does provide some advantages over version 1, there is still room for improvement.

The second modification, version 3, solves the difficulty inherent to version 2. When spurious rules are located, they are removed from both the rule set and the new entering set as in version 2. But in addition the rule is recorded on a list of undesirable rules. All incoming rules are checked against the undesirable list. If a match is found, that incoming rule is deleted. In this way a spurious rule, once found, is permanently prevented from reentering the system. While this process may cause a mild retardation in the learning rate due to the decreased number of rules used, the slowdown is more than compensated by the increased accuracy of the results. The workings of versions 1, 2, and 3 are summarized in Figure 10.

E) Experiments and Results

The experimentation consists of processing each of the three corpora with the three system versions, a total of nine runs in all. The corpora are each 180 sentences in length and are described previously in subsection B. The performance measures that are taken are shown in Figure 11. These results are tabulated in Figure 12. Figure 13 shows the resolution recall and precision for each word calculated at ten document intervals. Averages for the results in Figure 13 are presented in Figure 14. These results show how the overall system performance improves as more inputs are seen, thus indicating a true learning process. These charts also show the general superiority of version 3 over the other two. To indicate this fact more clearly, Figure 15 shows the difference in resolution recall and precision for the three versions averaged over all corpora. Version 1 is taken as the standard and lies on the x axis. Displacement above or below the x axis represents superiority or inferiority relative to version 1. These graphs show that version 2 and especially version 3 improve both resolution recall and precision over version 1. That is, not only do they perform more correct analyses than version 1, they also perform fewer incorrect analyses. Usually

INPUT	STATUS AFTER INPUT			
	V-1 Rule Set*	V-2 Rule Set*	V - 3 Rule Set	 Undesirable Rule List
A	LARGE TYPE	LARGE TYPE	LARGE TYPE	-
	(1)**	(1)	(1)	
B	LARGE TYPE	-	-	LARGE TYPE
	(2)**			
A	LARGE TYPE	LARGE TYPE	-	LARGE TYPE
	(1)	(1)	-	LARGE TYPE
	(1)	(1)		

* This chart shows only the part of the rule set that is relevant to this discussion.

** Numbers in parentheses indicate the interpretation associated with the rule.
Interpretation 1 is printing
Interpretation 2 is kind or variety

Summary of Versions 1, 2, and 3

Figure 10

T	The total number of ambiguities in the data set
C	The number of correctly resolved ambiguities
I	The number of incorrectly resolved ambiguities
U	The number of unresolved ambiguities
RR	Resolution Recall = C/T
RP	Resolution Precision = C/(C+I)

Performance Measures

Figure 11

WORD	VERSION	T	C	I	U	RR	RP
DEGREE	1	180	155	19	6	.86	.89
DEGREE	2	180	158	14	8	.88	.91
DEGREE	3	180	160	12	8	.89	.93
TYPE	1	180	166	10	4	.92	.94
TYPE	2	180	166	7	7	.92	.96
TYPE	3	180	164	4	12	.91	.98
VOLUME	1	180	144	30	6	.80	.83
VOLUME	2	180	144	30	6	.80	.83
VOLUME	3	180	152	15	13	.84	.91
TOTALS	1	540	465	59	16	.86	.89
	2	540	468	51	21	.87	.90
	3	540	476	31	33	.88	.94

General Results of Learning Process

Figure 12

DEGREE

NO. OF INPUTS PROCESSED	VERSION 1		VERSION 2		VERSION 3	
	RR	RP	RR	RP	RR	RP
10	.40	.80	.40	.80	.40	.80
20	.55	.79	.50	.77	.50	.77
30	.60	.75	.57	.77	.57	.77
40	.67	.79	.65	.81	.65	.81
50	.64	.72	.66	.79	.66	.79
60	.66	.74	.68	.79	.70	.81
70	.70	.77	.71	.81	.73	.82
80	.71	.77	.74	.82	.75	.83
90	.73	.79	.77	.84	.78	.85
100	.76	.81	.79	.86	.80	.87
110	.78	.83	.80	.86	.82	.88
120	.80	.84	.82	.87	.83	.89
130	.82	.85	.83	.89	.85	.90
140	.83	.86	.84	.89	.86	.91
150	.83	.87	.85	.90	.87	.92
160	.84	.88	.86	.91	.87	.92
170	.85	.88	.87	.91	.88	.93
180	.86	.89	.88	.92	.89	.93

Recall and Precision Results at Ten Input Intervals

Ambiguous word is DEGREE

Figure 13A

TYPE

NO. OF INPUTS PROCESSED	VERSION 1		VERSION 2		VERSION 3	
	RR	RP	RR	RP	RR	RP
10	.50	.71	.50	.71	.50	.71
20	.65	.76	.65	.81	.65	.81
30	.67	.77	.67	.83	.70	.88
40	.72	.81	.73	.85	.75	.88
50	.78	.85	.78	.89	.76	.90
60	.82	.88	.82	.91	.80	.92
70	.84	.89	.84	.92	.83	.94
80	.86	.91	.86	.93	.85	.94
90	.88	.92	.88	.94	.84	.95
100	.89	.93	.89	.95	.86	.96
110	.89	.92	.89	.94	.87	.96
120	.89	.92	.90	.95	.88	.96
130	.90	.93	.90	.95	.88	.97
140	.91	.93	.91	.95	.89	.97
150	.91	.94	.91	.96	.89	.97
160	.91	.94	.91	.95	.90	.97
170	.92	.94	.92	.96	.91	.97
180	.92	.94	.92	.96	.91	.98

Recall and Precision Results at Ten Input Intervals

Ambiguous word is TYPE

Figure 13B

VOLUME

NO. OF INPUTS PROCESSED	VERSION 1		VERSION 2		VERSION 3	
	RR	RP	RR	RP	RR	RP
10	.10	.17	.20	.33	.20	.33
20	.30	.40	.35	.47	.45	.64
30	.40	.50	.43	.54	.53	.70
40	.47	.56	.50	.59	.55	.67
50	.54	.61	.54	.61	.60	.71
60	.58	.65	.60	.67	.65	.75
70	.61	.67	.63	.69	.69	.79
80	.65	.70	.65	.70	.73	.82
90	.68	.73	.68	.73	.76	.84
100	.71	.76	.70	.74	.78	.86
110	.73	.77	.72	.76	.80	.87
120	.74	.78	.73	.77	.79	.87
130	.76	.80	.75	.78	.80	.88
140	.77	.81	.76	.80	.81	.89
150	.78	.81	.77	.81	.83	.90
160	.79	.82	.79	.82	.83	.90
170	.79	.82	.79	.82	.84	.90
180	.80	.83	.80	.83	.84	.91

Recall and Precision Resutls at Ten Input Intervals

Ambiguous word is VOLUME

Figure 13C

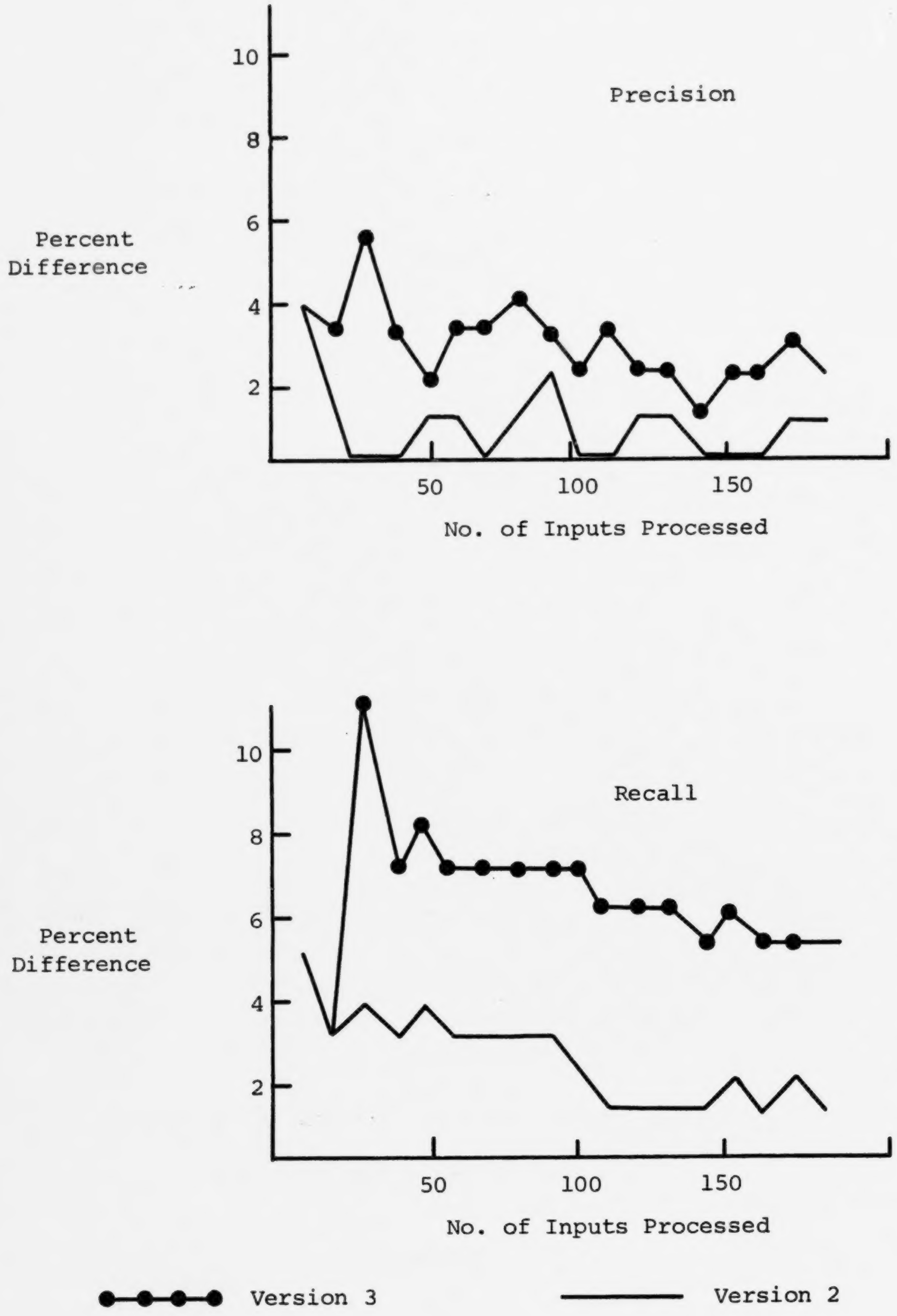
AVERAGES

NO. OF INPUTS PROCESSED	VERSION 1		VERSION 2		VERSION 3	
	RR	RP	RR	RP	RR	RP
10	.33	.56	.37	.61	.37	.61
20	.50	.65	.50	.68	.53	.74
30	.55	.67	.55	.71	.60	.78
40	.62	.72	.62	.75	.65	.79
50	.65	.72	.66	.76	.67	.80
60	.69	.76	.70	.79	.72	.83
70	.72	.78	.72	.81	.75	.85
80	.74	.79	.75	.82	.78	.86
90	.76	.81	.78	.84	.79	.88
100	.79	.83	.79	.85	.81	.90
110	.80	.84	.80	.85	.83	.90
120	.81	.85	.82	.86	.83	.91
130	.82	.86	.83	.87	.84	.92
140	.84	.87	.84	.88	.85	.92
150	.84	.87	.84	.89	.86	.93
160	.85	.88	.85	.89	.87	.93
170	.85	.88	.86	.90	.88	.93
180	.86	.89	.87	.90	.88	.94

Average Recall and Precision for All Corpora

Tabulated at Ten Input Intervals

Figure 14



Average Improvement Achieved by Versions 2 and 3
Over Version 1

Figure 15

this results in an increased number of unanalyzed inputs. This is actually a very desirable result since if a choice must be made between an input being analyzed incorrectly or not analyzed at all, the latter seems preferable. An example of this can be seen in Figure 13B. Version 2 produces only a few more correct analyses than does version 1, and thus the recall results show very little difference. However version 2 produces many fewer incorrect analyses thus significantly improving the precision results.

The results shown so far are prejudiced downward by the inclusion of the start-up portion of the learning process which necessarily performs poorly. Therefore a more important measure of system performance is a moving average. Figure 16 shows for each word the number of disambiguations performed correctly, incorrectly, and unanalyzed for each ten sentence group. These charts clearly indicate the anticipated poor start, the gradual improvement, and the final stabilization at near perfect performance. A 10 in the "Correct" column represents perfect resolution for that sentence group. These statistics are summarized by Figure 17. And in Figure 18, these averages are shown graphically. The x axis is the interval number. Interval 5, for example, contains inputs 41-50, etc. The y axis represents the number of correct analyses out of a possible 10. These charts are very graphic proof that the learning process builds and stabilizes at a high performance level.

Several other statistics are worthy of note. Figure 19 shows for each run the number of spurious templates and context rules contained in the rule sets at the end of that run. This number is broken down to show how many of these spurious rules occur in the first, middle, and last third of their respective rule sets. These figures indicate first that most rules learned by the system are not spurious; and secondly, that spurious rules

DEGREE INPUTS	VERSION 1			VERSION 2			VERSION 3		
	C	I	U	C	I	U	C	I	U
1-10	4	1	5	4	1	5	4	1	5
11-20	7	2	1	6	2	2	6	2	2
21-30	7	3	0	7	2	1	7	2	1
31-40	9	1	0	9	1	0	9	1	0
41-50	5	5	0	7	3	0	7	3	0
51-60	8	2	0	8	2	0	9	1	0
61-70	9	1	0	9	1	0	9	1	0
71-80	8	2	0	9	1	0	9	1	0
81-90	9	1	0	10	0	0	10	0	0
91-100	10	0	0	10	0	0	10	0	0
101-110	10	0	0	9	1	0	10	0	0
111-120	10	0	0	10	0	0	10	0	0
121-130	10	0	0	10	0	0	10	0	0
131-140	10	0	0	10	0	0	10	0	0
141-150	9	1	0	10	0	0	10	0	0
151-160	10	0	0	10	0	0	10	0	0
161-170	10	0	0	10	0	0	10	0	0
171-180	10	0	0	10	0	0	10	0	0

C	No. of Correct Analyses out of a Possible 10
I	No. of Incorrect Analyses
U	No. Unanalyzed

Disambiguation Performance for Ten Input Groups

Ambiguous word is DEGREE

Figure 16A

TYPE INPUTS	VERSION 1			VERSION 2			VERSION 3		
	C	I	U	C	I	U	C	I	U
1-10	5	2	3	5	2	3	5	2	3
11-20	8	2	0	8	1	1	8	1	1
21-30	7	2	1	7	1	2	8	0	2
31-40	9	1	0	9	1	0	9	1	0
41-50	10	0	0	10	0	0	8	0	2
51-60	10	0	0	10	0	0	10	0	0
61-70	10	0	0	10	0	0	10	0	0
71-80	10	0	0	10	0	0	10	0	0
81-90	10	0	0	10	0	0	8	0	2
91-100	10	0	0	10	0	0	10	0	0
101-110	9	1	0	9	1	0	10	0	0
111-120	9	1	0	10	0	0	10	0	0
121-130	10	0	0	9	0	1	8	0	2
131-140	10	0	0	10	0	0	10	0	0
141-150	10	0	0	10	0	0	10	0	0
151-160	9	1	0	9	1	0	10	0	0
161-170	10	0	0	10	0	0	10	0	0
171-180	10	0	0	10	0	0	10	0	0

C	No. of Correct Analyses Out of a Possible 10
I	No. of Incorrect Analyses
U	No. Unanalyzed

Disambiguation Performance for Ten Input Groups

Ambiguous word is TYPE

Figure 16B

VOLUME

INPUTS	VERSION 1			VERSION 2			VERSION 3		
	C	I	U	C	I	U	C	I	U
1-10	1	5	4	2	4	4	2	4	4
11-20	5	4	1	5	4	1	7	1	2
21-30	6	3	1	6	3	1	7	2	1
31-40	7	3	0	7	3	0	6	4	0
41-50	8	2	0	7	3	0	8	1	1
51-60	8	2	0	9	1	0	9	1	0
61-70	8	2	0	8	2	0	9	0	1
71-80	9	1	0	8	2	0	10	0	0
81-90	9	1	0	9	1	0	10	0	0
91-100	10	0	0	9	1	0	10	0	0
101-110	9	1	0	9	1	0	10	0	0
111-120	9	1	0	9	1	0	7	1	2
121-130	10	0	0	9	1	0	9	0	1
131-140	9	1	0	10	0	0	10	0	0
141-150	9	1	0	9	1	0	10	0	0
151-160	9	1	0	10	0	0	9	0	1
161-170	9	1	0	9	1	0	9	1	0
171-180	9	1	0	9	1	0	10	0	0

C	No. of Correct Analyses out of a Possible 10
I	No. of Incorrect Analyses
U	No. Unanalyzed

Disambiguation Performance for Ten Input Groups

Ambiguous word is VOLUME

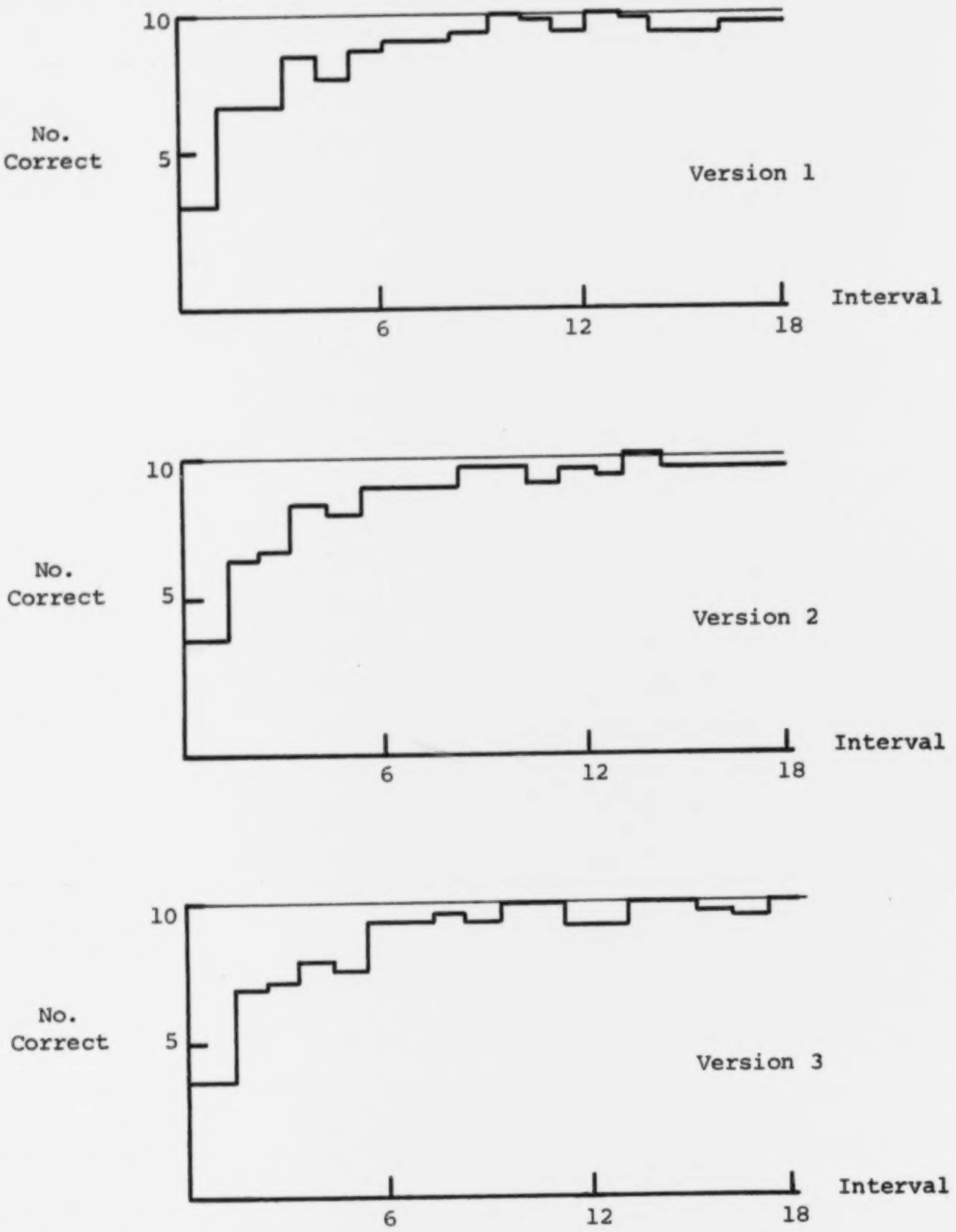
Figure 16C

INPUTS	AVERAGE NO. OF CORRECT ANALYSES OUT OF POSSIBLE 10		
	VERSION 1	VERSION 2	VERSION 3
1-10	3.33	3.67	3.67
11-20	6.67	6.67	7.00
21-30	6.67	6.67	7.33
31-40	8.33	8.33	8.00
41-50	7.67	8.00	7.67
51-60	8.67	9.00	9.33
61-70	9.00	9.00	9.33
71-80	9.00	9.00	9.67
81-90	9.33	9.67	9.33
91-100	10.00	9.67	10.00
101-110	9.97	9.00	10.00
111-120	9.33	9.67	9.00
121-130	10.00	9.33	9.00
131-140	9.97	10.00	10.00
141-150	9.33	9.67	10.00
151-160	9.33	9.67	9.97
161-170	9.67	9.67	9.67
171-180	9.67	9.67	10.00

Average Number of Correct Analyses for Each Ten Input Group

Maximum is 10

Figure 17



Average Number of Correct Analyses
For Each Ten Input Group

Figure 18

RUNS	# OF TEMPS	SPURIOUS				# OF C.R.	SPURIOUS			
		1/3	2/3	3/3	TOT		1/3	2/3	3/3	TOT
DEGREE V-1	31	2	2	7	11	25	0	3	4	7
DEGREE V-2	28	2	1	5	8	23	0	3	3	6
DEGREE V-3	19	1	0	1	2	22	0	1	5	6
TYPE V-1	21	1	3	4	8	15	0	3	3	6
TYPE V-2	18	1	3	2	6	12	0	2	1	3
TYPE V-3	20	1	3	3	7	10	0	1	1	2
VOLUME V-1	36	4	6	9	19	22	0	2	3	5
VOLUME V-2	31	3	3	8	14	21	0	2	2	4
VOLUME V-3	25	2	3	5	10	18	0	1	3	4
TOTAL	229	17	24	44	85	168	0	18	25	43

Number of Spurious Rules Found in the First, Middle and Last Third of Each Rule Set (For Each Run).

Figure 19

tend to be densest at the bottom of the rule sets. Thus due to the top-down search strategy, correct rules are far more likely to be chosen than spurious ones.

As stated previously one requirement for a good learning system is that it not be prone to unlearning. An input is considered to be unlearned if it is seen once and analyzed correctly and subsequently seen again and analyzed incorrectly. Figure 20 shows the number of unlearned inputs for each of the nine experimental runs. The low values here clearly indicate that once the system has learned to disambiguate a particular input, that capability remains learned. Also, the fact that versions 2 and 3 perform better than version 1 with respect to unlearning indicates that the prevention of spurious rules is an aid in the prevention of unlearning. Unlearning may stem from sources other than the system itself. If a user provides incorrect information to a learning system, improper rules and subsequent unlearning may result. In an operational learning system it may therefore be necessary for an analyst to review periodically the newly learned rules prior to their final acceptance into the permanent rule set.

One final investigation is to look at the contents of the undesirable rule lists following each version 3 run. Figure 21 shows the total number of rules in the lists and the number which by hand analysis are found to be actually spurious. Ideally all rules in these lists should be spurious; and the figures shown are quite close to this ideal. These results show that the system is able to learn not only the rules which make good disambiguaters, but also those which are not useful. The results presented here show these processes are truly capable of learning to disambiguate with a high degree of success.

F) Extensions

There are numerous other applications for a learning technique such

WORD	VERSION 1	VERSION 2	VERSION 3
DEGREE	1	1	0
TYPE	1	1	0
VOLUME	4	3	2
AVERAGE	2	1.67	0.67

Number of Unlearned Inputs for Each Run

Figure 20

RUN	USELESS TEMPLATE LIST		USELESS C.R. LIST	
	LENGTH	# SPUR	LENGTH	# SPUR
DEGREE	10	9	2	2
TYPE	1	1	1	1
VOLUME	9	8	3	3
TOTAL	20	18	6	6
ACCURACY	90%		100%	

Composition of the Useless Rule Lists

(Version 3 Only)

Figure 21

as the one presented previously. A large system with many users may be able to learn the individual needs and techniques of its users. The system could thus tailor a specialized subsystem to each individual. In the area of information retrieval a system might be able to learn to modify techniques and parameters in order to improve relevance feedback performance for a particular collection and user. In nearly any application where a set of rules or parameters must be created in order to perform some form of analysis, the learning technique is potentially valuable, especially where many such sets must be created to meet the needs of many users.

The learning process can also be applied to natural language analysis in the resolution of pronouns. Unlike ambiguities which have multiple meanings, pronouns have no meaning in isolation. To determine meaning, the word to which the pronoun refers must be located. This could be accomplished in the following way. The learning process looks at each noun in the vicinity of the pronoun and learns their contexts. These are then compared with the context of the pronoun and the noun with the best match used. There are of course some problems to be solved. For example, not all pronouns refer to a specific thing. The fact that some pronouns encompass large concepts or merely provide an impersonal subject can be seen in the second and third example sentences below.

A. Take an egg and break it into a bowl.
(specific reference)

B. The consequence of this is that the project is feasible.
(multiple reference)

C. These results show that it is possible.
(impersonal)

This can provide a more accurate natural language analysis process and improve performance in any natural language application.

6. Conclusion

This study is intended first to demonstrate the importance of disambiguation in various forms of natural language analysis, and to motivate investigation into the automation of this process. It also serves as a test of the template analysis facility. The study shows that it is possible to perform this disambiguation with a high degree of accuracy using an extended form of template analysis and a predetermined set of structured templates and unstructured context rules. The creation of these rules requires an analyst to examine typical inputs and determine the words or structures which indicate the intended meaning of the ambiguous word. As is shown in 5 this manual process may be eliminated by implementation of a process which allows the system to disambiguate for itself. With the exception of the first few inputs for which the performance is understandably low, the learning process demonstrates the same high degree of accuracy achieved with the hand made disambiguation rules. Not only is the system able to learn which rules provide good disambiguation, it can also determine which rules do not, and exclude these rules from the system. The learning process has applications in many areas and template analysis appears sufficiently general to facilitate many of the applications.

References

[1] Borillo, A., and Virbel, J., Resolution des Polysemies Dans L' Indexation Automatique de Resumes, Centre National de la Recherche Scientifique, Section D' Automatique Documentaire, June, 1966.

[2] Coyaud, M., Resolution of Lexical Ambiguities in Ophthalmology, Cornell University, Ithaca, New York, 1968.

[3] Dimsdale, B., and Lamson, B. G., A Natural Language Information Retrieval System, Proceedings of the IEEE, Vol. 54, No. 12, December 1966.

[4] Douglas, E., Mandersloot, W., and Spicer, N., Thesaurus Control — the Selection, Grouping, and Cross-referencing of Terms for Inclusion in a Coordinate Index Word List., Journal of the American Society for Information Science, January - February, 1970.

[5] Harris, Z. S., From Phoneme to Morpheme, Language, Vol. 31, No. 2, 1955.

[6] Kellogg, C. H., A Natural Language Compiler for On-line Data Management, AFIPS Conference Proceedings, Vol. 33, Proc. AFIPS 1968 Fall Joint Computer Conference, Vol. 33, Thompson Book Company, Washington, D. C.

[7] Lesk, M. E., A Proposal for English Text Analysis, Bell Telephone Laboratories, 1969.

[8] Moyne, J. A., A Progress Report on the Use of English in Information Retrieval, IBM Corporation, Federal Systems Center, Gaithersburg, Maryland, June 1969.

[9] Quillian, M. R., The Teachable Language Comprehender: A Simulation Program and Theory of Language, CACM Vol. 12, No. 8, August 1969.

[10] Rangathan, S. R., Recall Value and Entry Word in Headings, Library Science, Vol. 6, No. 4, December 1969.

[11] Simmons, R. F., Answering English Questions by Computer: A Survey, CACM, Vol. 8, No. 1, January 1965.

[12] Simmons, R. F., Synthex. In Orr, W.D., (Ed.), Conversational Computers, John Wiley and Sons, Inc., New York, 1968.

[13] Simmons, R. F., Natural Language Question-answer Systems: 1969, CACM, Vol. 13, No. 1, January 1969.

[14] Stone, P. J., Improved Quality of Content Computerized-disambiguation Rules for High Frequency English Words. In Analysis of Communication Content, Wiley, New York, 1969.

[15] Thompson, F. B. DEACON Type Query Systems. In Orr, W. D., (Ed.), Conversational Computers, John Wiley and Sons, Inc., New York, 1968.

[16] Weiss, S. F., A Template Approach to Natural Language Analysis for Information Retrieval, Ph.D. Thesis, Cornell University, Ithaca, New York, 1970.

V. The Effect of Common Words and Synonyms on Retrieval Performance

D. Bergmark

Abstract

The effect of removing common words from document and query vectors is investigated, using the Cran-200 collection. The method used is comparison of a standard stem dictionary and a thesaurus with a new dictionary formed by adding an extensive common word list to the standard stem dictionary. It is found that removal of common words from the query and document vectors significantly increases precision. Query and document vectors without either common words or synonyms yield the highest precision results but inferior recall rssults. Synonyms are found to be more effective for recall than common words.

1. Introduction

A thesaurus results in about 10% better retrieval than a standard stem dictionary, according to results in previous studies [2]. This fact leads to the question of why the thesaurus performs better: is it because it groups terms into synonym classes, or is it because the thesaurus includes a large common word list. If both contribute to the superiority of the thesaurus, then it is desirable to determine what proportion of this improvement is due to each factor. Taking common words out of a thesaurus could consume little time compared to that required for grouping concepts into synonym classes if an appropriate means of automatically generating the common word list were found. Therefore, if a large amount of improvement of a thesaurus over the stem dictionary is due to removing common words and putting them in a separate list, then it would be advantageous to devote work to methods of isolating the insignificant words.

The subject of this paper, then, is a comparison of the search results of a standard stem dictionary, a thesaurus, and a standard stem dictionary with an extensive common word list. The results of this study indicate that a large amount of the difference in retrieval performance between thesaurus and standard stem dictionaries is due to the removal of common words into a separate list. Surprisingly, the effect of synonyms and of common words are similar; both encourage higher recall but both degrade precision.

2. Experiment Outline

A) The Experimental Data Base

With limited resources, it is fairly important to choose carefully the collection to be studied. First, the collection must be small enough to be manageable within the resources available, yet large enough to give significant results. The collection also has to have both a thesaurus and a word stem dictionary available.

The Cran-200 collection seems to satisfy these criteria and is chosen as the basis for the study. This collection has 200 documents and 42 queries, and the text is available on tape for lookup with a new dictionary.

B) Creation of the Significant Stem Dictionary

Investigating the retrieval effectiveness of an extensive common word list together with a standard stem dictionary requires, per force, the generation of a new dictionary. Specifically, the new dictionary desired is one which has the same stems as the standard stem dictionary but with many more words marked as common.

The most readily available common word list for the Cran-200 collection is contained in the Cran-200 thesaurus. In fact, the thesaurus is essentially the same dictionary as the standard stem dictionary except that many more words are flagged as common, and synonyms are grouped into concept classes by assignment of the same concept number to all word stems synonymous with each other. Furthermore, since the same word may occur in more than one concept class, one term may have more than one concept number assigned to it.

Thus more "significance" decisions are made in constructing a thesaurus than in constructing a standard stem dictionary, both in removing common and in removing infrequently used words from the dictionary list. Hence if a thesaurus is turned back into a standard stem dictionary, the result is a standard stem dictionary with a large common word list. Therefore, rather than going through the standard stem dictionary and marking additional words as common, the strategy followed in this experiment is to go through the thesaurus and renumber the words so that the common words are still flagged as common, but the stems are separated so that no two stems have the same concept number and each stem has only one concept number. This method is efficient since no word-matching need be done to determine which are common words and which are not.

Punching the Cran-200 thesaurus, CRTHES, from Tape 9 onto cards yields approximately 3380 cards with one thesaurus term per card along with its concept class(es). These cards are then used as input to a 360/20 RPG program which punches a duplicate deck in which each thesaurus term is assigned a unique concept number, with numbering starting at 1 for the significant terms and at 32001 for common terms. This results in 2946 significant, distinct words and 741 distinct common words.

That the resulting dictionary (henceforth referred to as the "significant stem dictionary") is the one desired can be seen from Appendix I, which lists some typical query vectors using each of the three dictionaries. It can be seen that the significant and standard stem queries are sufficiently similar except for the inclusion of common words in the standard stem queries.* The significant stem dictionary has approximately twice as many words marked as common than does the standard stem dictionary. In addition, the significant stem dictionary has about 65% as many significant concepts as the standard, and many of the remainder are actually common and so were never included, or were deleted from, the thesaurus. The new dictionary thus has the same word significance decisions (i.e., the same common word list) as the thesaurus, but the same grouping decisions (i.e., none) as the word stem dictionary.

C) Generation of New Query and Document Vectors

With the creation of the new dictionary, it is necessary to reassign vectors for the queries and documents of the Cran-200 collection in preparation for search runs. To accomplish this task the LOOKUP program, written in PL/I, is used. This program reads in a dictionary, a suffix list, and the query or document texts; it then generates concept vectors for the texts using the standard suffixing rules. It is run once for the queries and once for the documents.

Some decision has to be made concerning the suffix list; ideally it should be as close as possible to that used for creating the original thesaurus and standard stem vectors for the Cran-200 collection. The suffix list used in this study contains approximately 195 terms, and the resulting vectors indicate that it is quite similar to the one used to generate thesaurus and standard stem vectors.

---

*There was some concern in the early stages of this work that the thesaurus contains many full words rather than stems. Although there are full words in the thesaurus which are only stems in the stem dictionary, the reverse is also true. In any case, analysis of individual queries shows that these discrepancies have no significant effect on what is retrieved.

As far as the Cran-200 text is concerned, it has to be picked out from the Cran-1400 collection. A slight modification of the LOOKUP program does this by allowing the user to specify which of the Cran-1400 query and document texts are to be processed. One Cran-200 text (Text 995) is not on the Cran-1400 tape but is fortunately not relevant to any of the Cran-200 queries; it is not believed that the missing document perturbs results very much.

The average length of the resulting significant stem queries is 6.14 words as opposed to the standard stem queries with 8.26 words and the thesaurus queries with 6.98 words. The size of the document vectors varies proportionally with the length of the queries, except that the thesaurus document vectors are in general slightly shorter than the significant stem document vectors.

Why there are more words in the thesaurus queries than in the significant stem queries is somewhat unclear. As can be seen from the queries listed in Appendix I the additional words in the thesaurus queries are common ones; these words have been removed from the thesaurus, probably because they were judged to be common, and thus do not appear in the significant stem queries. On the other hand, some thesaurus queries have fewer significant terms than the significant stem queries; this is because if two words are synonymous, their concept number appears only once in the thesaurus query with a heavier weight.

D) Document Analysis — Search and Average Runs

In order that the evaluation of all three dictionaries is on a consistent basis, search runs must be done using vectors generated with all three dictionaries. Relevancy judgments must be added to the significant stem query vectors obtained by LOOKUP so that the same relevancy judgments are used for each of the three sets of queries. A fairly simple search without complex parameters is performed so that unnecessary complications in analysis do not arise. A full search lists the top thirty documents, and then a positive feedback search using the top five documents is done to make sure that removing common words and synonyms does not have an unforseen effect on feedback.

The results of the three searches, thesaurus, significant stem and standard stem, are compared by analysis of overall measures as well as in-depth analysis of individual queries to see to what extent not having synonyms hurt or help the retrieval process. Similarly, in-depth analysis is required to see what effect common words, or lack of them, have on retrieval.

To aid the analysis, the standard averages are obtained as well as the recall-level and document-level recall-precision graphs. The three full searches are compared with each other, and the three feedback runs are compared with each other. Results are verified using the standard significance tests.

In addition, some statistics are calculated by hand to determine retrieval effectiveness. Specifically, it is felt that the default rank recall measure provided in the SMART averaging routines is not quite suited to the analysis being done here. When some of the relevant documents do not have any correlation with the query, the averages have to be based on extrapolation; in the standard SMART run, the rank recall is calculated assuming that the relevant documents with no correlation appear at the bottom of the list (i.e., rank 200, 199, 198,...). Since this project is directed toward seeing what effect common words have on precision as well as recall, it seems better to take into account the number of documents, relevant and non-relevant, which correlate with the query in the first place. That is, it seems that if one is testing precision, and if two queries each retrieve six out of nine relevant documents, but one of them recovers thirty more non-relevant documents than the other before going on to a zero correlation, it should be judged less precise than the other. Thus in the graphs derived by hand, rank recall is extrapolated on the basis of CORR.RANK+1, CORR.RANK+2, etc. for the relevant documents which have zero correlation with the query.

All in-depth analysis is performed on the full search results rather than on feedback results because the project is more concerned with determining the effect of dictionaries rather than the effect of feedback on retrieval. The recall-precision graphs for the three feedback runs are, however, included in Appendix II.

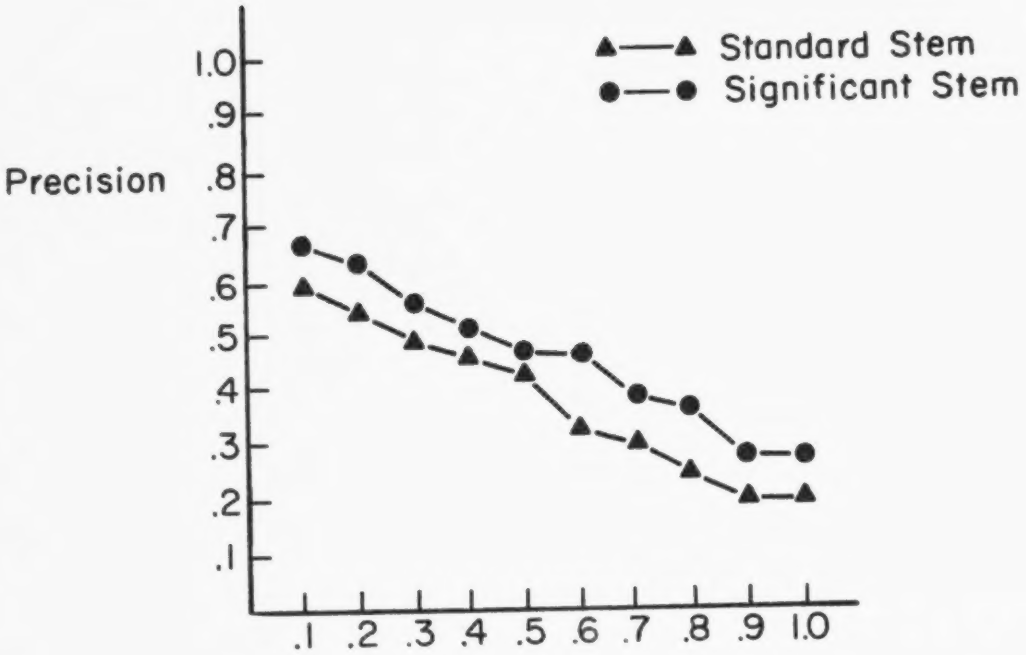
3. Retrieval Performance Results

A) Significant vs. Standard Stem Dictionary

The results of this experiment show that, as expected, use of a large common word list improves the retrieval performance of a standard stem dictionary. It can be seem from Graphs 1 and 2, which show the recall and precision averages for two full searches, one using the standard stem dictionary and the other using the significant stem dictionary, that the significant stem dictionary results in greater precision at all recall and document levels.

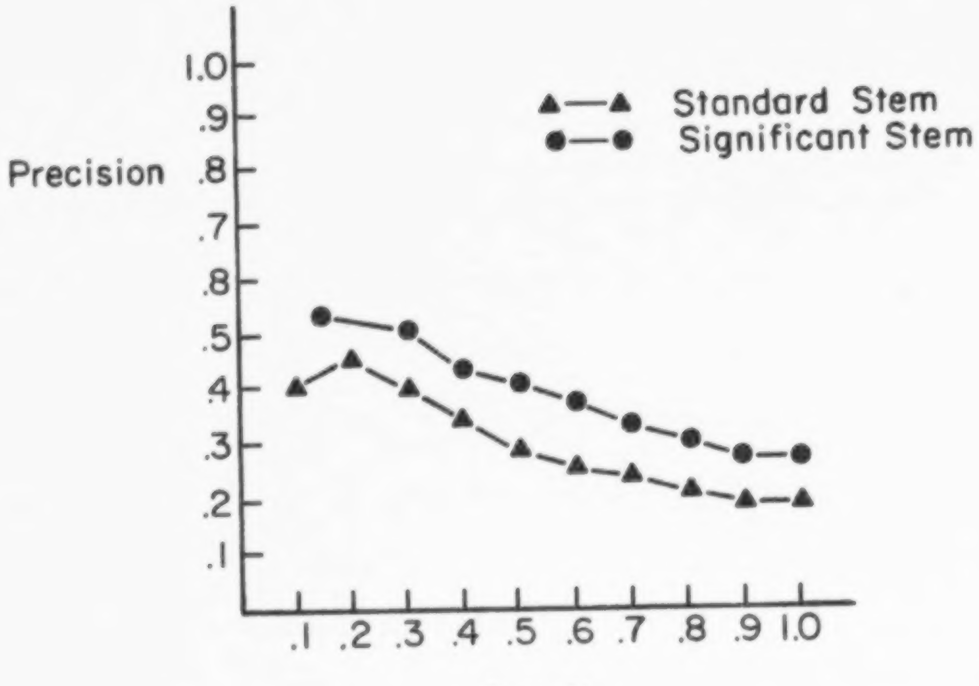
Furthermore, global statistics for these runs bear out the same conclusion, that the significant stem performs better than the standard stem:

	Standard Stem	Significant Stem
Rank Recall	.2424	.3331
Log Precision	.4202	.5053



Significant vs. Standard Stem
Recall-Level Averages
Full Search

Graph 1



Significant vs. Standard Stem
Document-Level Averages
Full Search

Graph 2

The above statistics are significant according to all the usual significance tests.

It is interesting to note that the difference between the significant and standard stem curves remains fairly constant despite the recall or document level. This indicates that the significant stem performs roughly the same retrieval as the standard stem, only more precisely. In other words, including common terms in the document and query vectors seems to uniformly degrade precision performance.

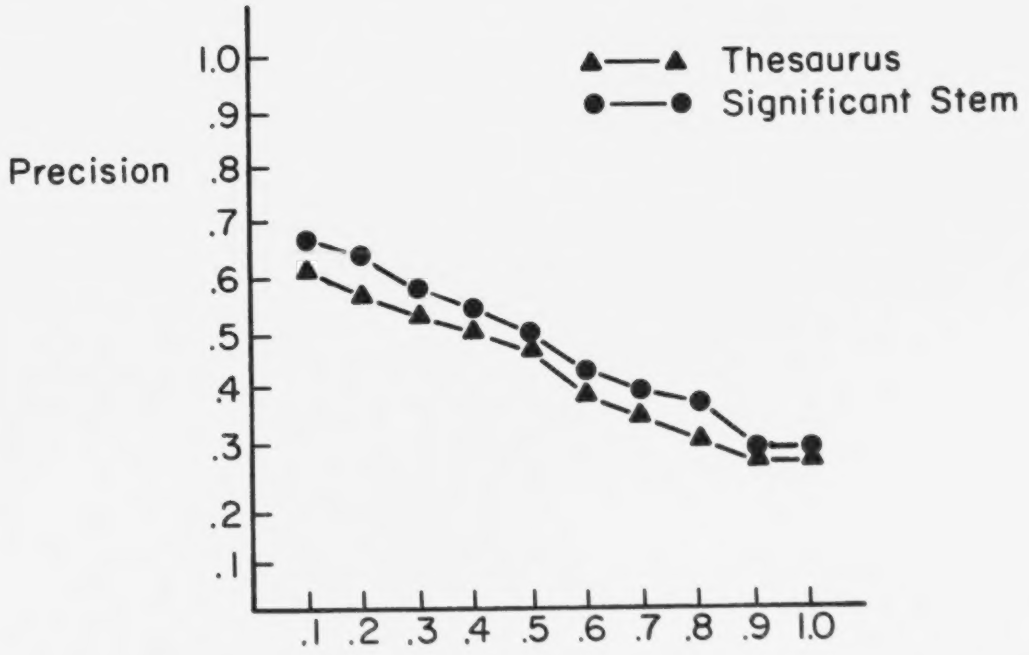
B) Significant Stem vs. Thesaurus

It was originally expected ttat using a standard stem dictionary with a large common word list would result in search performance better than the standard stem but not as good as the thesaurus. From the recall-precision Graphs 3 and 4 it can be seen that contrary to these expectations the significant stem performs just as well as the thesaurus, if not better.

The similarity of the significant stem and thesaurus curves is confirmed by global statistics, which while extremely close give a slight edge to the significant stem dictionary:

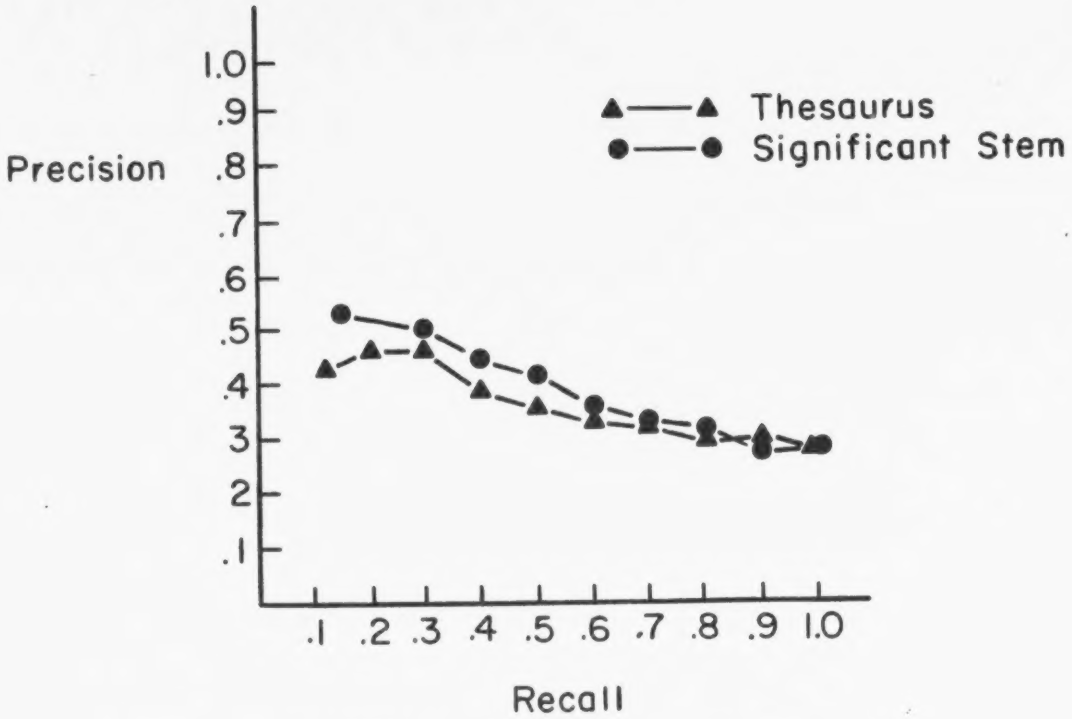
	Significant Stem	Thesaurus
Rank Recall	.3331	.3222
Log Precision	.5053	.4880

Here the difference between the two curves is not the same. The significant stem performs better than the thesaurus at the low end of the curve, but loses this edge as recall increases. One may conclude that the standard stem queries find only the first few relevant documents faster than



Significant Stem vs. Thesaurus
Recall-Level Averages
Full Search

Graph 3



Significant Stem vs. Thesaurus
Document-Level Averages
Full Search

Graph 4

the thesaurus.

C) Standard Stem vs. Thesaurus

In general a thesaurus results in better retrieval performance than a standard stem dictionary, and this experiment has roughly the same appearance. Recall-Precision Graphs 5 and 6 indicate the superiority of the thesaurus over the standard stem at all recall and document levels, with the superiority most marked at high recall levels. That the thesaurus, with its common word list and synonyms, is better than the standard stem but is approximately equal to the significant stem, with only a common word list, indicates that much of the improvement of the thesaurus over the standard stem is due to the common word list. Furthermore, comparison of these three sets of recall-precision plots seems to indicate that at the low recall end synonyms actually degrade precision, acting as common words do.

D) Recall Results

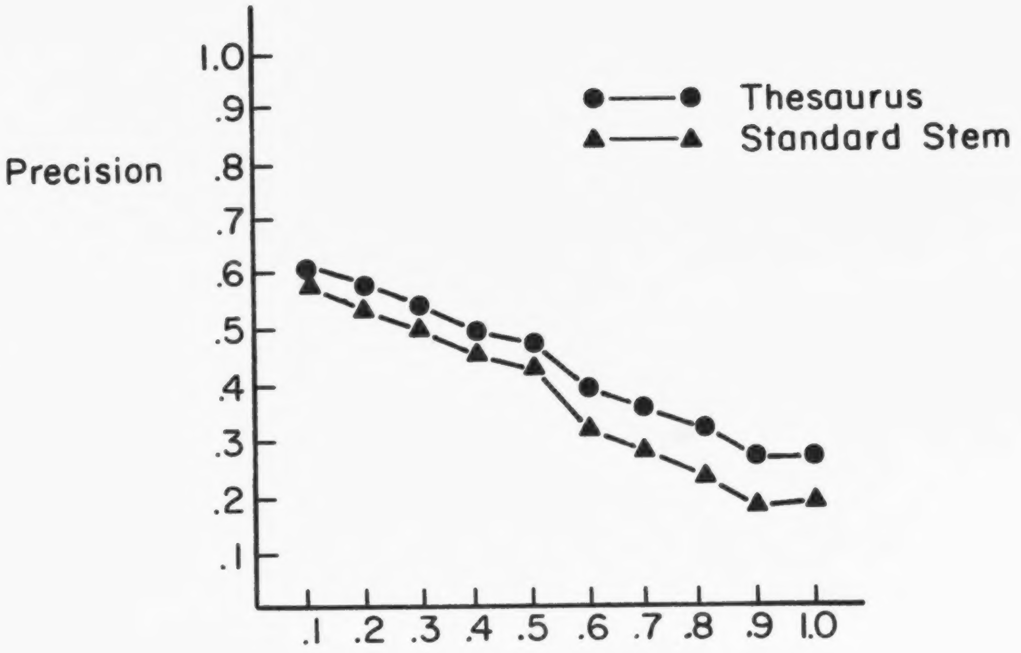
The difficulty with the significant stem dictionary, however, can be detected in the normalized global statistics (Figure 1).

	Standard Stem	Significant Stem	Thesaurus
Norm Recall	.8489	.8330	.8732
Norm Precision	.6615	.6918	.6924

Normal Recall and Precision for Full Search, All Dictionaries

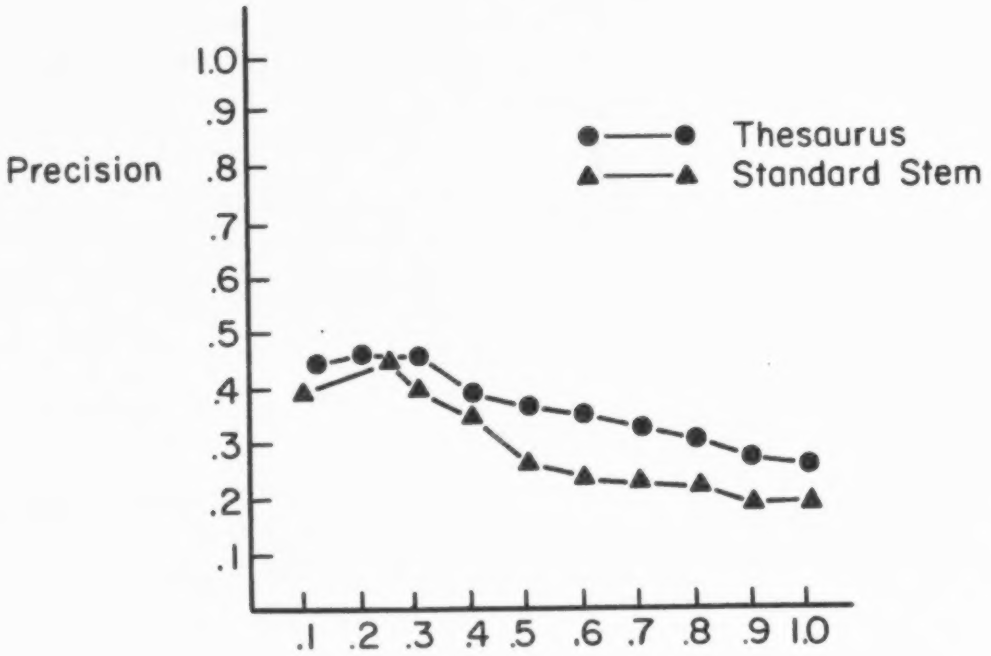
Figure 1

These global statistics are much closer than the Rank Recall and Log Precision and indeed, the first favors the standard stem dictionary over the significant stem although neither are significantly different according



Thesaurus vs. Standard Stem
Recall-Level Averages
Full Search

Graph 5



Thesaurus vs. Standard Stem
Document - Level Averages
Full Search

Graph 6

to the t-test. The problem displayed here is that the significant stem ultimately results in lower recall than does the standard stem; more queries have rank and precision measures based on extrapolation in the first case than in the second.

To be specific, 14 of the 42 queries using the significant stem dictionary do not have a 1.00 recall ceiling during the full search, while only nine of the standard stem and six of the thesaurus do not. The average recall ceiling for the significant stem is 0.8853 while the average recall ceiling for the standard stem is 0.9390 and 0.9565 for the thesaurus. After feedback, however, the difference narrows somewhat, going to 0.9504 for the significant stem dictionary and 0.9841 for the standard stem dictionary (the thesaurus at 0.9814 after feedback is not quite as good as the standard stem dictionary).

It is reasonable that the recall ceiling is higher for the standard stem than for the significant stem, since the average query length for the latter is greater than that for the former. Thus chances for a significant stem query not correlating at all with documents relevant to it are greater than those for a standard stem query. Similarly synonyms improve the chances for the thesaurus query's matching at least one relevant document.

To measure this recall difference in another way, Figure 2 displays a recall measure used by Keene [2]based on the average rank of the last relevant document retrieved. Figure 2 is based on the full search results.

The method 1 averages, which measure ultimate recall ability, shows that the thesaurus is superior in this respect, while the significant stem dictionary has the poorest recall. The method 2 averages, however, which are more a measure of precision in that they also include a measure of how many non-relevant documents are retrieved before correlation goes to zero,

Dictionary	Method 1	Method 2
Standard Stem	83.33	60.29
Significant Stem	87.64	46.45
Thesaurus	73.24	57.57

Method 1: Unrecovered relevant documents assigned ranks of 200, 199, etc.

Method 2: Unrecovered relevant documents assigned ranks of CORR.RANK+1, CORR.RANK+2, etc. where CORR.RANK is the rank of the documents with the lowest correlation with the query greater than 0.

Average Rank of the Last Relevant Document

Figure 2

put the significant stem at the top of the list. Thus these averages reinforce the previous hypothesis that if the user wants to recover every last relevant document, he should use the thesaurus, and if instead he is interested in minimizing the number of non-relevant retrieved, he should use the significant stem dictionary.

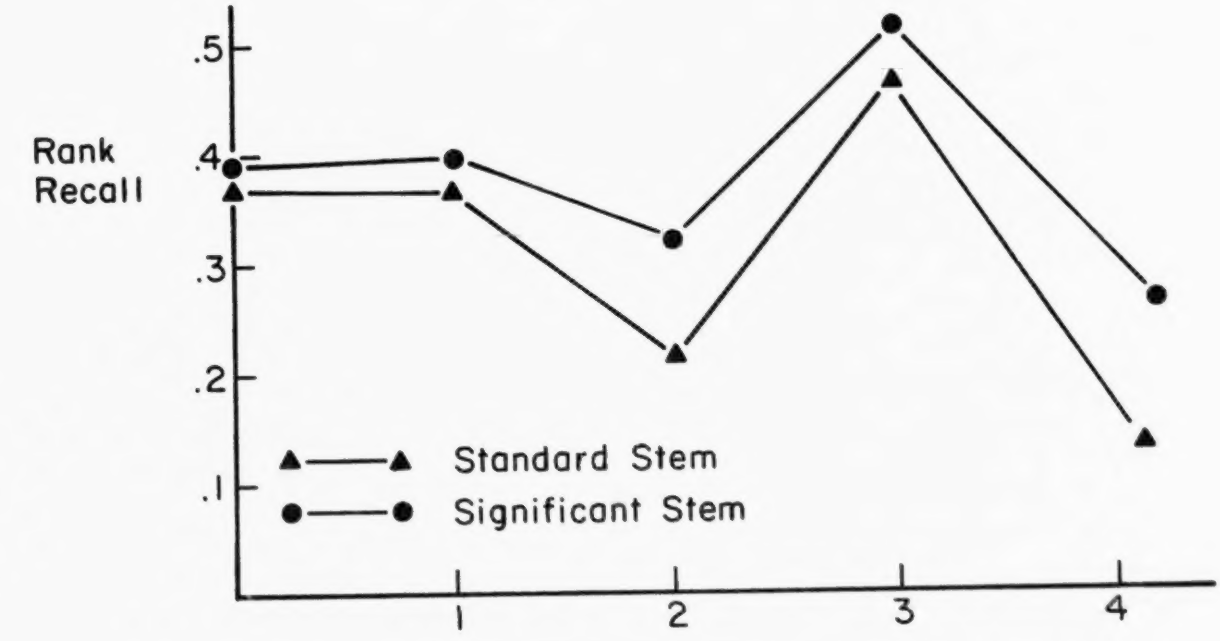
E) Effect of "Query Wordiness" on Search Performance

While it seems clear that significant stem results in an overall increase in precision over standard stem queries, it seems likely that the "wordiness" of a query, or the number of common words included in the standard stem query not included in the significant stem query, should have some effect on retrieval. That is, the more verbose a standard stem query is, the more non-relevant documents should be retrieved before all the relevant ones. Graph 7 shows the rank recall averages for standard and significant stems, over all 42 queries, at various levels of "wordiness".

It is not really clear that retrieval deteriorates faster as more and more common words are added to the query. A couple of possible explanations for this are 1) all the common words together may retrieve the same documents, since the common words in a given query may be "related", or 2) of the common words added, only one or two of them are responsible for retrieving garbage. (The latter theory seems to be confirmed by study of individual queries.) The left part of the graph is of course identical for both dictionaries since at that point the queries are practically identical.

F) Effect of Query Length on Search Performance

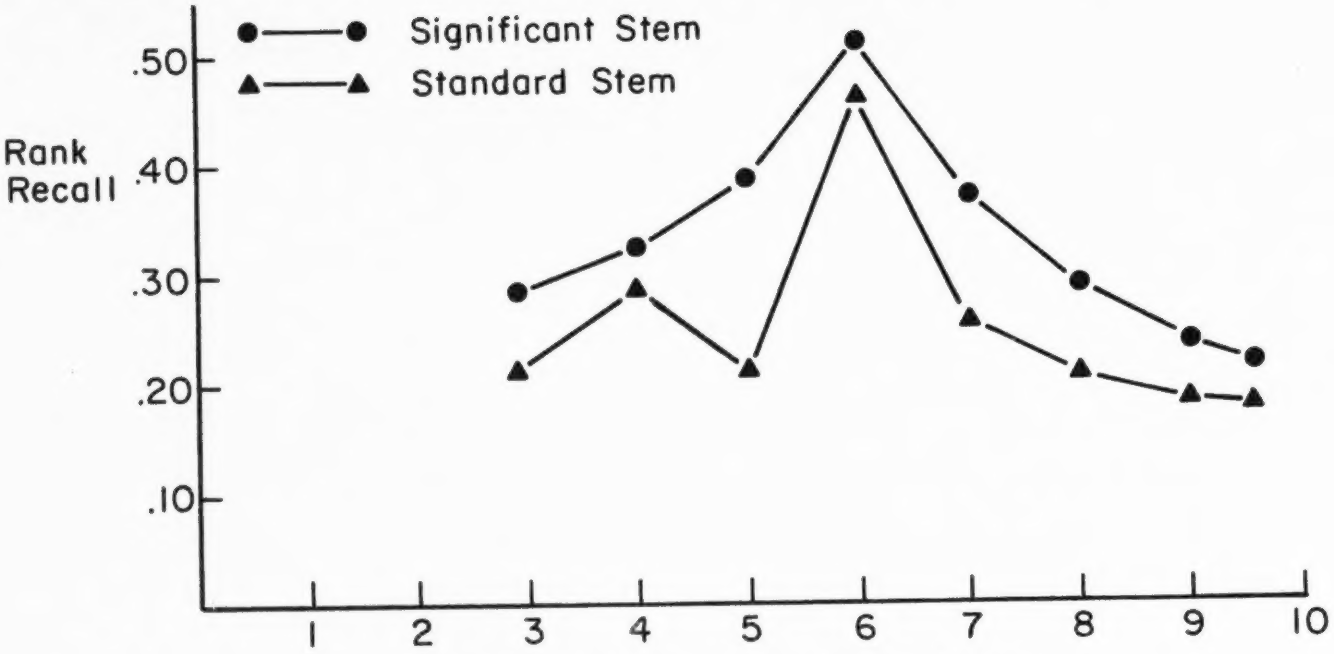
It also seems likely that the difference in performance would vary depending on the number of significant concepts in the query. For example, if the significant stem query is very explicit, containing many significant



# of Common Words in Standard Stem

Rank Recall vs. Wordiness

Graph 7



Significant Concepts in Query

Length of Query vs. Rank Recall

Graph 8

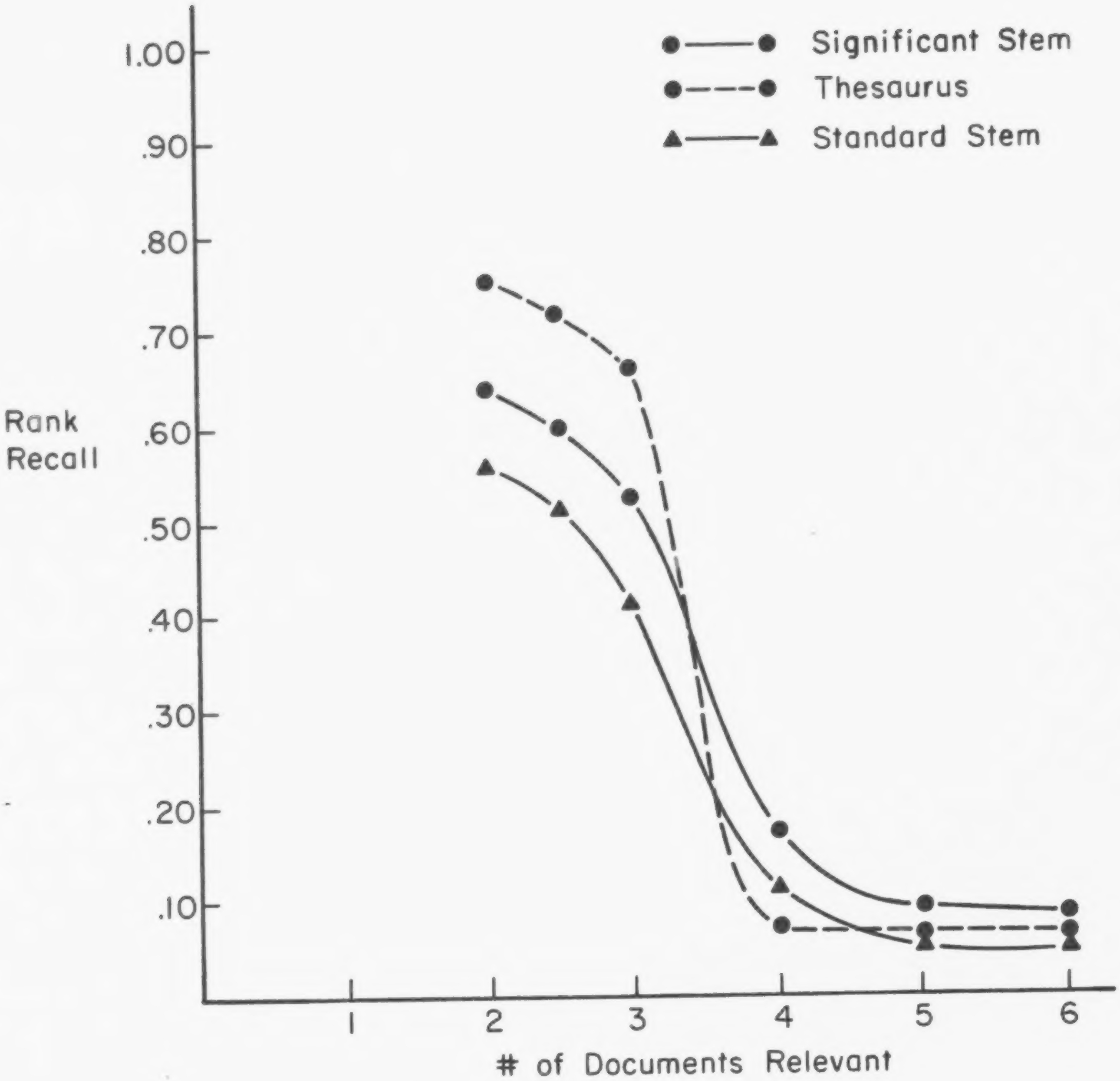
concepts in it, then the added common words in the standard stem query should result in extremely precise retrieval. On the other hand, a very short query in terms of significant concepts would, one supposes, almost have to contain common words if any documents are to be retrieved at all. This hypothesis, however, is not born out by the search results. Graph 8 plots rank recall for the significant and standard stem queries at various query lengths over 42 queries.

Graph 8 indicates that there are indeed differences in the improvement of significant stem over standard stem queries, but there is no easy way to characterize the differences. There are other factors affecting retrieval, such as the number of documents relevant to the query. For example, with a very short query and few relevant documents, common words would be more necessary than if there are a lot of relevant documents. Thus the only fact shown by Graph 8 is that retrieval can vary with the length of the query; the best recall occurs at the average number of significant concepts, which is roughly six.

G) Effect of Query Generality on Search Performance

Remaining is the question of whether it is wise to forget about using a thesaurus with synonyms, since removing common words alone improves stem retrieval. Certainly the recall-precision graphs indicate that precision suffers with the thesaurus, particularly at low recall and document levels. In many cases, then, it appears that synonyms retrieve more non-relevant documents than a dictionary without synonyms.

Graph 9, however, indicates that the picture for the thesaurus is not all that black. This graph shows, for all three dictionaries, rank recall plotted against the number of documents relevant to the query, holding query length constant; when query generality is low, the thesaurus performs best.



Rank Recall vs. # Documents Relevant
(Queries with 6 Significant Concepts)

Graph 9

Using a thesaurus improves the chances of those one or two relevant documents being retrieved, whereas the signficant stem query may fail to correlate with any of the relevant documents. When there are many relevant documents, however, a thesaurus loses its edge because at least one of the relevant documents is likely to be retrieved by any of the queries, and the thesaurus synonyms serve only to retrieve a large amount of non-relevant items.

H) Conclusions of the Global Analysis

The general conclusions which may be drawn from this global analysis are as follows:

1) If one is interested in precision, it is definitely wise to remove common words from the query and document vectors.
2) If one is interested in a high recall ceiling during a full search, one should use a thesaurus. The thesaurus has better ultimate recall than does stem alone, indicating that synonyms retrieve better than common words do.
3) If there are few documents relevant to a query, one should use a thesaurus. Keen reaches much the same conclusion, saying that "for users needing high precision with only one or two relevant documents, the thesaurus is little better than stem on IRE-3, but in CRAN-1 and ADI, a larger superiority for the thesaurus is evident." [2] (CRAN-1 is the same collection as is being used here.) It is possible that while synonyms are useful in the Cran-200 and ADI collections, in other collections synonyms would not be required even for high recall.
4) If there are many relevant documents to a query, it is just as good and perhaps better to remove both common words and synonyms from the query and document vectors.

4. Analysis of Search Performance

Having reached some conclusions on the basis of overall statistics, it is now appropriate to examine the reasons for these results by looking at some specific queries.

The overall averages presented in section 3 indicate the general superiority of the significant stem dictionary over the standard stem dictionary. At all recall (and document) levels, the significant stem has greater precision than does the standard stem. The reason for this improvement in performance can be seen by inspection of Query 36 (Figure 3).

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
37	1	.4234	1	.5292	1	.4889
35	2	.2413	2	.3111	2	.3651
36	7	.1365	4	.2046	6	.2614
34	14	.1064	5	.1519	5	.2505
Rank Sum	.4167		.8333		.7143	
Log Precision	.4503		.8615		.7762	
Norm Recall	.8941		.9974		.9949	
Norm Precision	.7843		.9716		.9491	

Query 36

Figure 3

The standard stem query has two more terms in it than does the significant stem query, "determine" and "establish." It can be seen from Figure 3 that removal of these two common words from the query doubles search effectiveness.

All three queries retrieve documents 35 and 37 first; the standard stem query, however, retrieves four non-relevant documents before the third relevant one. Two of these non-relevant documents are retrieved by the query word "determine" while the other two are retrieved simply because they are short and contain one query term each.

Analysis of this query demonstrates two reasons why removing common words is beneficial to retrieval. One is that common words increase the chances of the query's correlating with a non-relevant document simply because that document and the query have the same common words in them. Secondly, inclusion of common words greatly increases the length of the document vectors, but short texts are lengthened relatively less than are long texts. Thus short texts have a decidedly greater chance of a high correlation with the query; having one term in common with the query gives it a disproportionately high correlation when relevancy should not be a function of text length.

Also indicated by the recall-precision curves is the similarity of the significant stem and thesaurus retrieval, with the significant being slightly better in general. This finding is also borne out by Query 36 (Figure 3), where only two non-relevant documents are retrieved by the thesaurus query, as opposed to the one retrieved by the significant stem query, before a recall level of 1.00 is reached. Interestingly, the short document containing the terms "axial compressor" which was retrieved early by both the stem queries is not one of these two non-relevant documents retrieved early by the thesaurus query; rather, synonyms account for the retrieval of the two non-relevant items. Specifically, the query term "compressor" appears only once in the two non-relevant documents, while the synonym "impeller" appears seventeen times, giving them a high correlation with the thesaurus query.

Query 36 thus demonstrates why synonyms can degrade precision; "compressor" is a frequently occurring word in the Cran-200 collection and in combination with its synonyms can cause retrieval of a number of non-relevant documents. Using stems alone, on the other hand, gives less emphasis to words like "compressor" and more to the group of significant query terms as a whole.

Nevertheless, it is difficult to make hard and fast distinctions between the search precision of thesaurus queries versus significant stem queries. In Query 27 (Figure 4), for example, it is precisely the synonyms which account for the high performance of the thesaurus query. All three versions of Query 27 are identical, except that the thesaurus query, of course, includes synonyms. These synonyms serve to retrieve with relatively high precision the first three relevant documents. Specifically, document 160 does not contain the term "boundary-layer" but it does contain its synonyms "boundary" and "layer" three times each. In this case, the low precision effect of synonyms is offset by the large set of query terms; taken as a whole, the complete set of query terms and their synonyms helps pinpoint the relevant documents more accurately.

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
160	45	.1826	34	.2287	5	.4327
28	43	.1902	46	.2020	8	.3813
56	31	.2105	32	.2297	11	.3750
29	75	.1035	77	.1226	54	.2307
71	62	.1284	57	.1667	71	.1405
161	138	.0309	-	-	166	.0367
Norm Recall	.6796		.3333		.7285	
Norm Precision	.2920		.3754		.4772	
Rank Recall	.0533		.0150		.0623	
Log Precision	.2699		.1692		.3336	

Query 27

Figure 4

The superior correlation of relevant items 28 and 56 with the thesaurus query as opposed to the stem queries is explained by the shorter thesaurus document vector lengths (Figure 5).

Document	Thesaurus Length	Significant Stem Length
28	57	63
56	26	27

Length of Relevant Document Vectors for Query 27

Figure 5

Similarly, the significant stem is more precise than the standard stem because significant stem document vectors are shorter, giving higher weights to their significant terms.

Search results in this study corroborate the findings of past workers that the thesaurus is better than the standard stem dictionaries. The results also indicate that much of this difference may well be attributable to the lengthy common word list of the thesaurus. In Query 36 (Figure 3), for example, the improvement of the thesaurus query over the standard stem query is due more to the removal of common words than to synonyms.

The same improvement can be seen in Query 7 (Figure 6) where the thesaurus results in much better retrieval than the standard stem query. All three queries retrieve the same two relevant and the same non-relevant documents in the first three recovered. After that, however, the next relevant document is found in ranks 11, 13, and 41 for the significant stem, thesaurus, and standard stem queries, respectively. This difference in retrieval is clearly due to the removal of common words, since the two dictionaries with the long common word list ranked about the same. Synonyms

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
41	2	.4042	1	.4914	1	.4762
90	3	.3175	3	.3536	3	.3859
42	41	.1459	11	.2176	13	.2572
72	53	.1279	47	.1211	35	.1918
95	60	.1200	70	.0773	44	.1672
Norm Recall	.8523		.8800		.9169	
Norm Precision	.5944		.6856		.7130	
Rank Recall	.0943		.1136		.1563	
Log Precision	.3528		.4129		.4351	

Query 7

Figure 6

contribute very little to the high precision in the initial retrieval stages.

Results indicate, however, that at the higher recall levels, the thesaurus is superior. This is shown in Query 7 (Figure 6) where the last two relevant documents are retrieved much faster by the thesaurus query than by either of the two stem queries. The reason for this is primarily the shorter document lengths of the thesaurus vectors, and secondarily the synonym "coefficient" is matched with the query term "derivative" in one case. (Shorter document length also explains the faster retrieval of 72 by the significant stem than by the standard stem.) In the case of document 95, however, the standard dictionary works better than the significant stem dictionary because the common terms "comparison" and "number" combined with the significant "mach" boost the document-query correlation of 95.)

That the significant stem dictionary has severe short-comings in the lower correlation, high recall, ranges is without doubt. This degradation in recall is not fully reflected by the recall-precision graphs, though it is seen in the normalized global statistics (Figure 1).

The main explanation for this phenomenon appears to be that the significant stem vectors, with neither common words nor synonyms in them, have a good chance of "missing" a relevant document altogether. Query 23 (Figure 7) demonstrates this in that one of the two relevant documents does not correlate at all with the significant stem query.

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
143	3	.2197	5	.1257	10	.1991
148	13	.1346	-	-	5	.2683
Norm Recall	.9672		.4899		.9697	
Norm Precision	.6999		.3722		.6748	
Rank Recall	.1875		.0146		.2000	
Log Precision	.1892		.1003		.1772	

Query 23

Figure 7

In this query, Item 148 has none of the significant query terms. It does, however, contain the synonyms "impeller" and "Compressor" for the query term "pump," and it also contains "method," a common term found in the standard stem query. (It should be noted that Document 148 is picked up after feedback for the significant stem query.)

While both common words and synonyms are useful for retrieval at high recall levels, synonyms are superior in this respect. In Query 3 (Figure 8) the thesaurus is the only dictionary of the three which achieves 100% recall during the full search.

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
57	3	.2134	3	.2889	8	.3303
31	24	.1331	14	.1862	13	.2476
30	16	.1486	21	.1795	20	.2182
32	9	.1825	10	.2102	23	.2001
4	18	.1450	19	.1827	25	.1876
33	-	-	-	-	124	.0441
Norm Recall	.7861		.7887		.8351	
Norm Precision	.5681		.5724		.5132	
Rank Recall	.0778		.0787		.0986	
Log Precision	.3774		.3797		.3497	

Query 3

Figure 8

The only reason that document 33 is retrieved by the thesaurus is that it contains the term "high-pressure-ratio" which matches "pressure" in the thesaurus query. Even the five extra terms added to the standard stem dictionary query fail to retrieve this last relevant item.

It is interesting to note here that while recall is superior for the thesaurus in Query 3, precision is not. The synonyms, as noted above, retrieve many non-relevant documents, and here more so than even common words do. Once again, the rule that high recall means low precision seems to be borne out.

Although the significant stem fails to achieve a 100% recall ceiling more often than both the other dictionaries, there are cases when high precision, low recall, and feedback can be effectively used to achieve high precision and high recall. One case of this is Query 1 (Figure 9) where so many non-relevant items are retrieved by the thesaurus and the standard stem that feedback is impossible because the user sees no relevant documents. Once again, as is typically the case, the thesaurus has the highest recall ceiling but not very precise retrieval.

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
22	29	.0899	1	.2209	33	.1109
21	-	-	-	-	32	.1115
1	-	-	-	-	-	-
	Query 1 after feedback					
22	29	.0899	1	.9796	33	.1109
21	-	-	9	.0955	32	.1115
1	-	-	2	.1996	-	-

Query 1

Figure 9

The significant stem query retrieves only one of the three relevant items (22), but this item is used for positive feedback and in turn retrieves another relevant document (21). No feedback, on the other hand, can be done with the standard stem query (only 22 correlates, and it is in rank 29) or with the thesaurus query (two relevant documents correlate with the query, but are in ranks 32 and 33). Thus query 1 demonstrates that it is not always necessary to have complete recall, at least during the initial search; high precision is more useful if feedback is going to be used.

The feedback recall-precision graphs in Appendix II indicate that this is precisely what happens, since feedback improves the precision of the significant stem much more than the other two dictionaries at the high recall end of the curve.

The effect of query length on precision, where length is the number of significant concepts in the query vectors, does not appear to vary retrieval results in a consistent manner. If a query is worded very specifically, which dictionary used is immaterial (see Query 12, Figure 10). On the other hand, a lengthy query may zero in faster on relevant documents but in the long run retrieves more non-relevant ones.

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
46	1	.5175	3	.5284	5	.5217
49	2	.4759	2	.5423	2	.7272
48	4	.4308	7	.4558	7	.4937
50	5	.3996	4	.5185	3	.6963
47	6	.3857	5	.4642	6	.5067
51	7	.3776	8	.4082	8	.4660
Norm Recall	.9966		.9931		.9914	
Norm Precision	.9663		.9111		.8950	
Rank Recall	.8400		.7241		.6774	
Log Precision	.8859		.7466		.7137	

Query 12
Figure 10

The length of the query is less important than the number of documents relevant to a query. If there are a lot of documents relevant to a query, it is often better to use a narrow query first (no common words or synonyms) and then use feedback to retrieve the remaining relevant items. In Query 16 (Figure 11) the thesaurus has the highest recall ceiling in the full search, but at the same time retrieves so many non-relevant that only one relevant item is available for feedback. The standard stem does not have quite a high recall ceiling and also has only one document in the top five for feedback. The significant stem, however, retrieves two relevant in the top five and so feedback is more effective (the total of relevant document ranks after feedback is the least for the significant stem query).

Relevant Document Number	Standard Stem Full	Standard Stem Feed-back	Significant Stem Full	Significant Stem Feed-back	Thesaurus Full	Thesaurus Feed-back
102	2	1	2	1	2	1
84	9	37	5	2	20	25
83	7	5	9	3	11	5
81	-	2	-	4	70	2
80	15	3	15	5	27	3
82	-	16	-	6	-	13
193	18	18	21	14	9	4
67	24	31	22	38	46	41
85	-	50	-	41	-	33
Sum of ranks after feedback		163		114		127

Query 16, Full Search and Feedback Rankings

Figure 11

It seems obvious, then, that an extensive common word list is helpful in retrieval, particularly if precision is desired. If one wishes to improve upon a standard stem dictionary, the first thing he should do is to find a good, extensive common word list. After that, additional improvement may be gained (in recall, particularly) by grouping some of the dictionary terms into concept clssses. Doing it the other way around can be disastrous, however, as is seen in Query 19 (Figure 12).

Relevant Document #	Standard Stem Rank & Corr.		Significant Stem Rank & Corr.		Thesaurus Rank & Corr.	
123	19	.2016	3	.3636	15	.2303
125	20	.1990	5	.3079	21	.2052
122	6	.2490	6	.2814	18	.2107
124	47	.1254	18	.1886	62	.1375
Norm Recall	.8954		.9719		.8648	
Norm Precision	.5327		.7658		.4667	
Rank Recall	.1087		.3125		.0862	
Log Precision	.2744		.4300		.2489	

Query 19

The significant stem dictionary here is clearly the best and the thesaurus is the worst. In Query 19, there are eight significant terms which in themselves result in good retrieval (as indicated by the performance of the significant stem query). In addition to these eight terms, there are five common terms in the standard stem query, causing it to retrieve five non-relevant items before the first relevant one. Figure 13 shows how the significant terms can be overwhelmed by insignificant terms.

Document	94	86	64	25	148	122 R
signif. terms, in all queries	planform rectangular wing	analytic flow oscillate rectangular wing	planform wing	analytic flow transonic	flow	flow oscillate transonic wing
common terms, in stand. stem only	determine general method possible	determine general method	general method	determine general method	determine general method	method

Terms (and Number of Occurrences) Appearing in Top 6
Documents Retrieved by Standard Stem Query 19

Figure 13

The thesaurus query vector for some reason contains three of the common terms added to Query 19; it does even worse than the stem dictionaries because synonyms compound the difficulties of common words. The thesaurus query thus retrieves 14 non-relevant documents before finding the first relevant one. The query terms "oscillater" and "planform" both belong to relatively large synonym classes.

5. Conclusions

The main conclusion of this study in the area of dictionary construction is that careful construction of common word lists is at least as important as grouping concepts into synonym classes. This is an important result since it should be earier to construct common word lists automatically than to construct synonym classes automatically.*

This study, in addition, has relevance to areas other than dictionary construction. For example, a fair amount of work is being done in the area of automatic document vector modification, which in part involves dropping "unimportant" concepts from the vectors (i.e., concepts infrequently used in queries). Since the common word list used in this study also contains infrequent words whereas the standard stem dictionary merely includes them as regular words, there is an opportunity in local analysis of these search runs to determine the effect of infrequently used words on retrieval. In particular both Query 6 and Query 1 in some of their versions included an infrequent word not in the other versions. In neither case, did this infrequent word affect retrieval except lower correlations by lengthening the query vector.

Another area in which this study is relevant is in scatter storage schemes for dictionary lookups [3]. This scheme can offer improvements in efficiency but thesaurus-type dictionaries are difficult to handle. One has to make a two-step mapping in order to get to the synonym class from the original query or document term; common words, on the other hand, can be handled easily enough. Therefore having determined that a standard stem dictionary can be considerably improved by removing some words into the common word list, it would be better to implement this improvement in the storage scatter scheme than it would be to implement the improvement involving concept classes.

---

* Work is being done in automatic synonym construction or has been done [1]. For these algorithms to work, however, common words probably have to be removed first, anyway.

Finally, this project carries out a suggestion made by Keen [2] that is the "five rules" of thesaurus construction are to be really evaluated, several different versions of a single dictionary would have to be made and tested. In the course of this study, a new dictionary is created, one which uses the frequency rules but not the grouping rules. Thus the importance of rules dealing with word frequency versus rules about synonym classes is established. It is just as important to be careful in constructing the common word list as in constructing the thesaurus. However, it is probably easier to follow the rules for common work list construction since common words are more systematic than synonyms are.

6. Further Studies

This investigation raises a few issues which were not settled, and which may prove interesting for further study:

1) The work presented in this paper is of course not conclusive for collections other than the Cran-200. The first extension of this experiment, then, would be to perform a similar common word analysis on other collections. One reason for the apparent good performance of the significant stem dictionary is that the Cran-200 thesaurus is not that much better than the standard stem dictionary in the first place.

2) The current Cran-200 collection still contains a fair number of common words in the thesaurus vectors although these same words have been marked common in the thesaurus itself. This could also explain the lack of performance of the thesaurus as compared with the significant stem dictionary. Thus a new look-up run should be made on the Cran-200 collection using the current version of the thesaurus to generate vectors without so many common words in them.

3) It would be interesting to determine more precisely the influence of infrequent words on retrieval.

4) More careful analysis of feedback results from this investigation should be made.

References

[1] R. T. Dattola and D. M. Murray, "An Experiment in Automatic Thesaurus Construction," Report No. ISR-13 to the National Science Foundation, Section VIII, Cornell University, Department of Computer Science, 1968.

[2] E. M. Keen, "Thesaurus, Phrase & Hierarchy Dictionaries," Report No. ISR-13 to the National Science Foundation, Section VII, Cornell University, Department of Computer Science, 1968.

[3] D. M. Murray, "A Scatter Storage Scheme for Dictionary Lookups," Report No. ISR-16 to the National Science Foundation, Section II, Cornell University, Department of Computer Science, 1969.

Appendix I

Some query vectors using the standard stem, significant stem and thesaurus

Query	Standard Stem		Significant Stem		Thesaurus	
1	4116	gas	863	gas	226	gas
	5087	kinetic	1139	kinetic	118	kinetic
	2086	Chapman-Enskog				
	2576	detail			275	results, solution
	7296	rigorous				
	9083	theo-			33	theory
2	1553	bound-	253	boundary	394	boundary
	2463	cylinder	484	cylinder	158	cylinder
	3392	flow	777	flow	389	flow
	5171	layer	1178	layer	394	layer
			1441	non-circular	151	non-circular
3	2666	dissociate	568	dissociate	89	dissociate
	3137	enthalpy	656	enthalpy	294	enthalpy
	3479	free	822	free	11	free
	4407	hypersonic	977	hypersonic	57	hypersonic
	6625	press-	1690	pressure	386	pressure
	8248	simulate	2019	simulate	194	simulate
	8546	stream	2202	stream	414	stream
	9306	tunnel	2419	tunnel	190	tunnel
	9725	wind	2588	wind	190	wind
	4305	high			47	high
	6558	possible				
	7113	realize			521	real, practical
	7249	respect				
	8234	significant				
4	2447	current	477	current	332	current
	2609	differ-	547	difference	105	difference
	3035	effect	610	effect	388	effect
	4258	heat	906	heat	276	heat
	5168	law	1176	law	270	law
	8465	stagnation-point	2152	stagnation-point	134	stagnation-point
	9238	transfer	2389	transfer	251	transfer
			2534	viscosity-temperature		
	9618	vortice	2548	vortic-	281	vortic-
	1218	analyses			31	analyses
	1334	assume			17	assume
	2641	discrepancy				
	6652	prime			44	prime ?
	7257	result				

Query	Standard Stem		Significant Stem		Thesaurus	
5	4407	hypersonic	977	hypersonic	57	hypersonic
	5171	layer	1178	layer	394	layer
	5239	line-	1217	linear	288	linear
	7289	reynold-	1866	reynolds	362	reynolds
	8184	shock	1982	shock	387	shock
			2534	viscosity-temperature		
	1218	analyses			31	analyses
	1334	assume			17	assume
	5321	low			46	low
	6196	number			384	number
	8358	solution				
6	1388	axial	164	axial	185	axial
	2226	compress-	372	compressor	202	compressor
	5090	kink	1140	kink	242	kink
	5239	line	1216	line	68	line
	5594	multi-stage	1402	multi-stage		
	8665	surge	2258	surge	149	surge
	3248	explain				
7	1102	aerodynamic	39	aerodynamic	137	aerodynamic
	2551	derivatives	525	derivative	429	derivative
	4407	hypersonic	977	hypersonic	57	hypersonic
	5348	mach	1269	mach	392	mach
	5441	measure	1319	measure	32	measure
	2207	compare				
	6196	number			384	number
	9086	theoretic			36	theoretical
	9764	work				
8	1102	aerodynamic	39	aerodynamic	137	aerodynamic
	2551	derivatives	525	derivative		
	3285	facility	715	facility	207	facility
	5441	measure	1319	measure	32	measure
	7353	run-	1899	running	289	running
	8208	short	2003	short	53	short
	9169	time	2356	time	9	time
	1084	adopted				
	1377	avail				
	5479	method				
9	1107	aerofoil	44	aerofoil	197	aerofoil
	2370	correct-	439	correction		
	5582	mount	1385	mount	55	mount
	9306	tunnel	2419	tunnel	190	tunnel
	9330	two-dimensional	2436	two-dimension-	104	two-dimension-
	9727	wind-tunnel	2589	wind-tunnel	190	wind-tunnel

Query	Standard Stem	Significant Stem	Thesaurus
10	3392 flow	777 flow	389 flow
	7019 quasi-conical	1761 quasi-conical	157 quasi-conical
	8480 state	2163 state	26 state
	6621 present		
	9083 theo-		33 theory
11	3392 flow	777 flow	389 flow
	5128 laminar	1152 laminar	94 laminar
	5543 model	1367 model	194 model
	6019 nature-	1410 natural	297 natural
	6386 parameter	1580 parameter	271 parameter
	9242 transit-	2392 transition	394 transition
	9306 tunnel	2419 tunnel	190 tunnel
	9316 turbulent	2426 turbul-	286 turbul-
	9725 wind	2588 wind	190 wind
	4566 influence		249 influence
12	1060 act-	24 action	250/249 action
	1139 air	63 air	165/228 air
	1192 altitude	92 altitude	184 altitude
	1348 atmosphere	151 atmosphere	228 atmosphere
	2712 drag	588 drag	135 drag
	4273 height	918 height	184 height
	6284 orbit	1534 orbit	460 orbit
	8024 satellite	1913 satellite	318 satellite
	8031 scale	1915 scale	43 scale
	2334 contract		
	9536 vary		239 adjust
13	2543 delta	516 delta	159 delta
	3392 flow	777 flow	389 flow
	8408 speed	2118 speed	253 speed
	8682 sweptback	2268 sweptback	50 sweptback
	9035 tapered	2298 taper-	498 taper-
	9253 transonic	2398 transonic	296 transonic
	9755 wing	2592 wing	223 wing
	2609 differ		239 adjust

Query Vectors for Three Dictionary Types

Run 0 — 42 Queries (Plus 0 Nulls) — Wordstem Feedback = Standard
A Full Search with One Iteration of Feedback Using Word Stem Dictionary
Run 1 — 42 Queries (Plus 0 Nulls) — Cranmine Feedl = Sig Stem
Full Search with One Iteration of Feedback using Stems with Common Words
Run 2 — 42 Queries (Plus 0 Nulls) — Thesaurus Feedback
A Full Search with One Iteration of Feedback

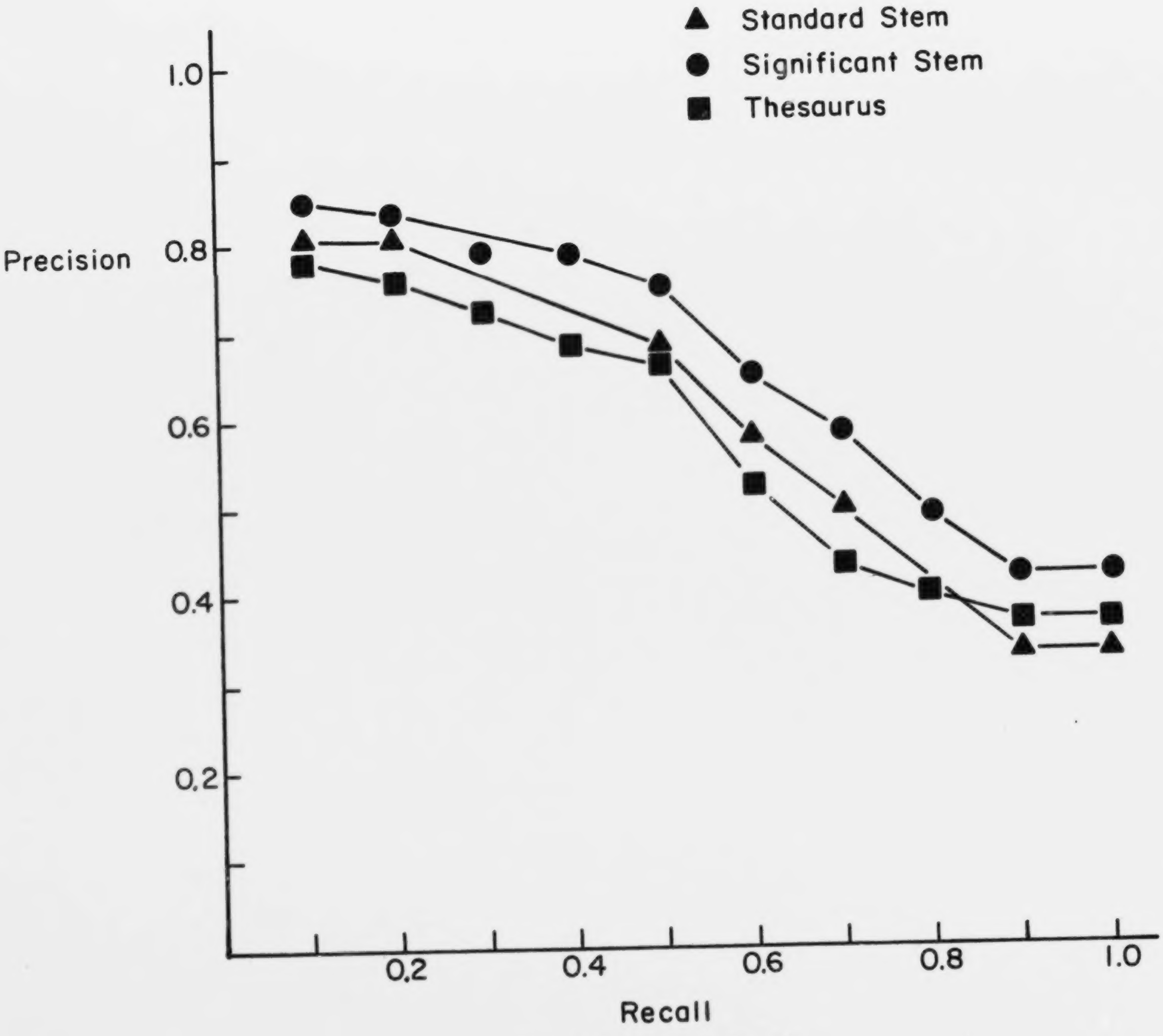
	Run 0		Run 1		Run 2	
Recall	NQ	Precision	NQ	Precision	NQ	Precision
0.0	0	0.8098	0	0.8484	0	0.7783
0.05	0	0.8098	0	0.8484	0	0.7783
0.10	1	0.8098	1	0.8484	1	0.7783
0.15	8	0.8098	8	0.8484	8	0.7664
0.20	21	0.8098	21	0.8246	21	0.7521
0.25	31	0.8098	30	0.8067	31	0.7291
0.30	31	0.7881	30	0.7885	31	0.7162
0.35	32	0.7710	32	0.7862	33	0.6983
0.40	32	0.7110	32	0.7862	33	0.6881
0.45	32	0.6810	32	0.7455	33	0.6597
0.50	40	0.6810	40	0.7455	41	0.6597
0.55	40	0.5883	40	0.6612	41	0.5503
0.60	40	0.5759	40	0.6479	41	0.5117
0.65	40	0.5234	40	0.6241	41	0.4757
0.70	40	0.4916	40	0.5799	40	0.4351
0.75	40	0.4698	40	0.5509	40	0.4347
0.80	40	0.4043	37	0.4831	39	0.3953
0.85	40	0.3736	34	0.4419	38	0.3734
0.90	40	0.3486	34	0.4278	38	0.3593
0.95	40	0.3486	34	0.4278	38	0.3580
1.00	41	0.3486	35	0.4278	39	0.3580
Norm Recall		0.8955		0.9011		0.9045
Norm Precision		0.7647		0.7999		0.7597
Rank Recall		0.4082		0.4997		0.4207
Log Precision		0.6001		0.6647		0.5885

Symbol Keys: NQ = Number of Queries used in the average not dependent on any extrapolation.
Norm = Normalized.

Recall Level Averages

Appendix 2

Recall Revision Results



Recall Level Graph

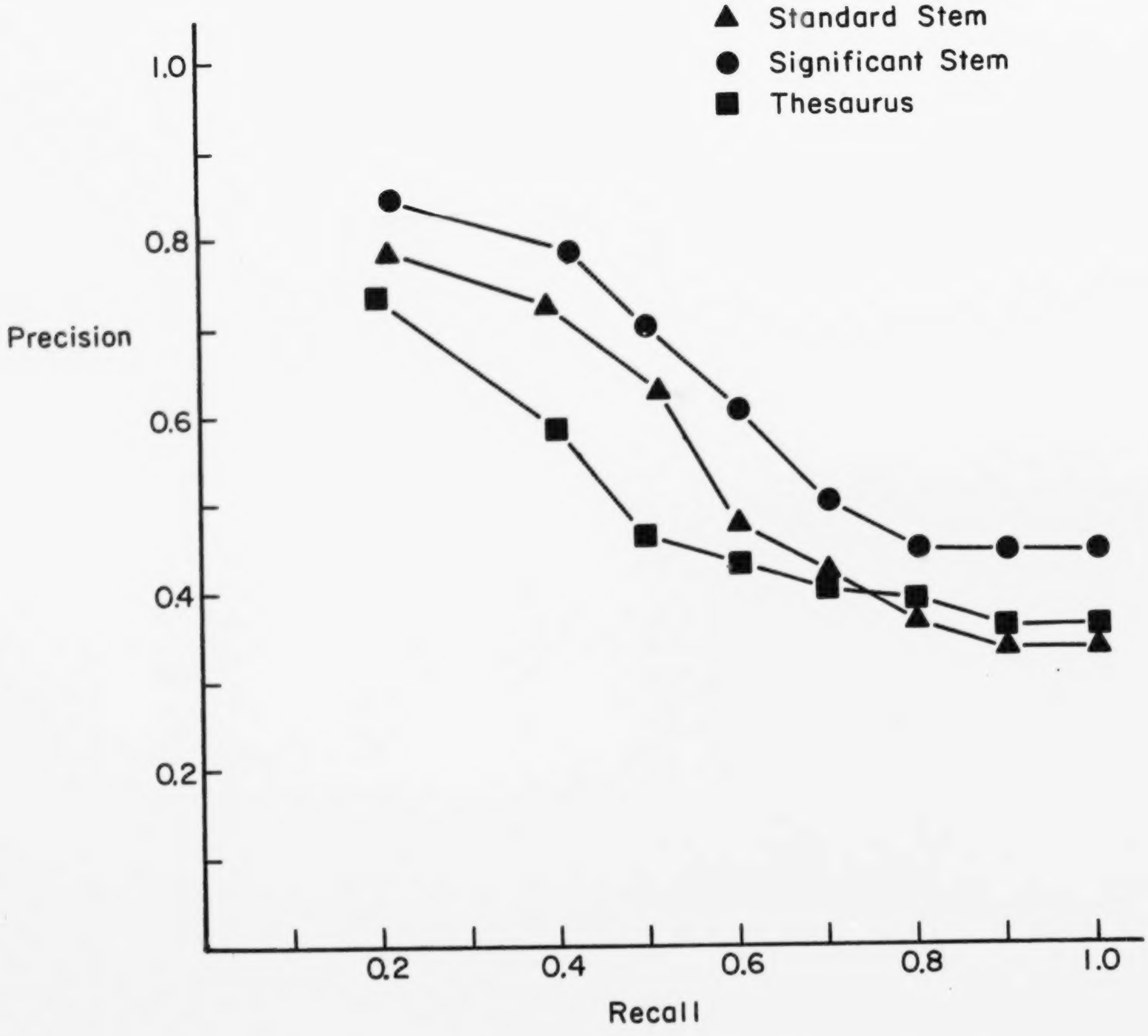
Run 0 — 42 Queries (Plus 0 Nulls) — Wordstem Feedback = Standard
A Full Search with One Iteration of
Feedback Using Word Stem Dictionary

RUN 0

Rank	NR	CNR	NQ	Recall	Precision
1	33	33	42	0.2266	0.7857
2	27	60	41	0.3817	0.7262
3	17	77	36	0.4555	0.6667
4	13	90	35	0.5129	0.6190
5	5	95	34	0.5293	0.5571
6	8	103	34	0.5651	0.5278
7	4	107	33	0.5798	0.4955
8	5	112	31	0.5993	0.4789
9	1	113	29	0.6033	0.4584
10	1	114	28	0.6072	0.4436
11	4	118	28	0.6287	0.4379
12	3	121	28	0.6416	0.4313
13	2	123	28	0.6485	0.4238
14	3	126	28	0.6622	0.4191
15	3	129	28	0.6749	0.4150
16	2	131	28	0.6805	0.4099
17	3	134	28	0.6921	0.4069
18	1	135	28	0.6947	0.4015
19	2	137	28	0.7054	0.3980
20	2	139	28	0.7148	0.3948
30	11	150	26	0.7612	0.3702
50	19	169	20	0.8448	0.3531
75	16	185	9	0.9321	0.3514
100	2	187	8	0.9395	0.3491
	11	198			
10.0%	139	139	28	0.7148	0.3948
25.0%	30	169	20	0.8448	0.3531
50.0%	18	187	8	0.9395	0.3491
75.0%	6	193	3	0.9683	0.3484
90.0%	1	194	2	0.9742	0.3483
100.0%	4	198	0	1.0000	0.3486

Symbol Keys:
NR = Number of Relevant.
CNR = Cumulative Number of Relevant.
NQ = Number of Queries used in the Average not Dependent on any Extrapolation.
% = Percent of Total Number of Items in Collection.

Document Level Averages (1)



Document Level Averages

Run 1 — 42 Queries (Plus 0 Nulls) — Cranmine Feedl = Sig Stem
Full Search with One Iteration of Feedback using Stems with Common Words

RUN 1

Rank	NR	CNR	NQ	Recall	Precision
1	35	35	42	0.2405	0.8333
2	28	63	41	0.4146	0.7619
3	18	81	35	0.5011	0.7063
4	12	93	32	0.5479	0.6528
5	9	102	31	0.5848	0.6111
6	8	110	31	0.6170	0.5794
7	5	115	29	0.6393	0.5510
8	5	120	27	0.6594	0.5349
9	3	123	26	0.6772	0.5170
10	2	125	23	0.6868	0.5038
11	2	127	22	0.6941	0.4940
12	4	131	21	0.7128	0.4912
13	4	135	20	0.7273	0.4893
14	2	137	20	0.7329	0.4843
15	2	139	20	0.7448	0.4800
16	2	141	19	0.7525	0.4767
17	1	142	19	0.7555	0.4723
18	1	143	19	0.7603	0.4684
19	0	143	19	0.7603	0.4637
20	1	144	19	0.7642	0.4606
30	12	156	18	0.8064	0.4429
50	20	176	11	0.8885	0.4355
75	6	182	6	0.9216	0.4310
100	4	186	2	0.9397	0.4291
	12	198			
10.0%	144	144	19	0.7642	0.4606
25.0%	32	176	11	0.8885	0.4355
50.0%	10	186	2	0.9397	0.4291
75.0%	2	188	0	0.9504	0.4275
90.0%	0	188	0	0.9504	0.4269
100.0%	10	198	0	1.0000	0.4278

Symbol Keys:
NR = Number of Relevant.
CNR = Cumulative Number of Relevant.
NQ = Number of Queries used in the Average not Dependent on any Extrapolation.
% = Percent of Total Number of Items in Collection.

Document Level Averages (2)

Run 2 — 42 Queries (Plus 0 Nulls) — Thesaurus Feedback
A Full Search with One Iteration of
Feedback

RUN 2

Rank	NR	CNR	NQ	Recall	Precision
1	31	31	42	0.2099	0.7381
2	24	55	41	0.3541	0.6667
3	10	65	36	0.3888	0.5714
4	15	80	36	0.4592	0.5536
5	6	86	34	0.4811	0.5060
6	4	90	34	0.5012	0.4663
7	8	98	34	0.5399	0.4515
8	9	107	33	0.5807	0.4452
9	6	113	29	0.6138	0.4389
10	2	115	28	0.6232	0.4254
11	6	121	27	0.6506	0.4239
12	3	124	25	0.6625	0.4186
13	4	128	25	0.6787	0.4160
14	1	129	25	0.6821	0.4087
15	2	131	24	0.6928	0.4047
16	1	132	24	0.6975	0.3998
17	3	135	24	0.7142	0.3982
18	2	137	23	0.7249	0.3958
19	2	139	23	0.7327	0.3936
20	3	142	23	0.7426	0.3929
30	15	157	22	0.7990	0.3777
50	18	175	15	0.8886	0.3662
75	10	185	10	0.9331	0.3616
100	0	185	10	0.9331	0.3583
	13	198			
10.0%	142	142	23	0.7426	0.3929
25.0%	33	175	15	0.8886	0.3662
50.0%	10	185	10	0.9331	0.3583
75.0%	9	194	2	0.9774	0.3580
90.0%	0	194	1	0.9774	0.3576
100.0%	4	198	0	1.0000	0.3580

Symbol Keys:
NR = Number of Relevant.
CNR = Cumulative Number of Relevant.
NQ = Number of Queries used in the Average not Dependent on any Extrapolation.
% = Percent of Total Number of Items in Collection.

Document Level Averages (3)

VI. Negative Dictionaries

K. Bonwit and J. Aste-Tonsmann

Abstract

A rationale for constructing negative dictionaries is discussed. Experimental dictionaries are produced and retrieval results examined.

1. Introduction

Information retrieval often involves language processing, and language processing frequently leads to language analysis. When the information initially appears in natural language form, it is desirable to perform some sort of normalization at the beginning of the analysis. A system often used in practice assigns keywords, or index terms, to identify the given information items. Dictionaries, listing permissible keywords and their definitions, are employed in this process. Sometimes, a negative dictionary is also used, to identify those terms which are not to be assigned as keywords.

Various types of positive dictionaries, their construction and uses, have been discussed elsewhere [1, 2, 3]. The question of the negative dictionary, or, what to leave out, is a fuzzy one. It is generally agreed that "common function words", such as "and", "or", "but", which add to the syntax but not the semantics of a sentence, should be dropped for the purposes of information retrieval. Other words at the extreme ends of the frequency distribution cause a problem. For example, "information" and "retrieval" might appear in nearly every document of a collection on that subject (high frequency); if included as keywords, they would retrieve everything. Conversely, if only one document discusses "microfiches" (low frequency), and that word does not constitute one of the permissible keywords, that document may never be retrieved. As with most information retrieval problems, the goals of the system, either high recall or high precision, will determine how many words are to be included. In the SMART system, a standard list of 204 "common English words" is used as a negative dictionary for all collections.

The general procedure used for dictionary construction consists in producing a concordance of the document collection with a frequency count, and including in the negative dictionary rare, low frequency words, common high frequency words, and words which appear in only nonsignificant contexts, such as "observe" in "we observe that . . ." This process requires the choice of frequency cutoff points, and a definition of the notion of "nonsignificance". It presumes a priori that such deletions will not effect retrieval results too considerably. A preferable system would be one that produces a negative dictionary of those terms which can be shown to detract from retrieval efficiency, or at least, not to affect it.

2. Theory

The set of keywords chosen for identifying documents constitutes the index language. The number and type of words included will control the specificity of the index language. Keen states [3] that

> "a dictionary which provides optimum specificity for a given test environment will exhibit a precision versus recall curve that is superior to all others probably over the whole performance range."

The purpose of this report is to exhibit a means of measuring specificity, and to show how a negative dictionary can be constructed to optimize index language specificity.

The aim of a negative dictionary is to delete from the index language all words which do not distinguish, and leave only those words which discriminate, among the documents. If the documents are considered as points in a vector space, with the associated identifying keywords as coordinates, then documents containing many of the same keywords will be relatively close together. If all keywords are permitted, then the documents will all cluster in the subspace defined by the common words; on the other hand, if only discriminators are permitted, the document space will "spread out", since each discriminator separates the space into those documents it identifies and those it does not.

The standard method for measuring "closeness", or correlation, of two document vectors $\underline{v}$ and $\underline{w}$ is the cosine:

$$\cos(\underline{v},\underline{w}) = \frac{\sum v_i \cdot w_i}{\sqrt{\sum v_i^2 \cdot \sum w_i^2}}$$

where v_i (w_i) is the weight of the i^{th} keyword in document $\underline{v}$ ($\underline{w}$), and the sums run over all possible keywords.

The "compactness" ("closeness together") of the points in the document space can be measured as follows:

1) find the centroid $\underline{c}$ of all the document points, that is,

$$c_i = \frac{1}{N} \cdot \sum_{j=1}^{N} v_{ij}$$

where v_{ij} is the weight of the i^{th} keyword in document j, and N is the total number of documents;

2) find the correlation of each document with the centroid, i.e., $\cos(\underline{c},\underline{v}_j)$, for all documents j;

3) define the document space similarity, Q, as:

$$Q = \sum_{j=1}^{N} \cos(\underline{c},\underline{v}_j)$$

Q has values between 0 and N, higher values indicating more similarity among documents. The value 0 is never obtained since $\underline{c}$ is a function of the other vectors, and the value N is obtained only if all the documents are identical to the centroid. Normalized Q, i.e. Q/N, is just the average document-centroid correlation (though this value is never calculated in the work which follows).

By calculating Q, using the terms provided by differing index languages, it is possible to measure and compare the specificity of these languages — a language is more specific the lower its Q. The question remains how to discover the optimal Q that will give the superior recall-precision curve described by Keen.

To see what happens when a single keyword is deleted, let Q_i be defined as Q calculated with the i^{th} term deleted (i.e., v_{ij} left out of all calculations, for all documents j). Then, $|Q - Q_i|$ measures the change in document space similarity due to the deletion of term i. If $Q_i > Q$, the document space is more "bunched up", more similar, when term i is deleted, or term i is a discriminator. Conversely, if $Q_i < Q$, deletion of term i causes the space to "spread out", to be more dissimilar, and deletion of term i may aid in retrieval. In the same way, Q_I is defined for a set of terms,

$I = \{i_1, i_2, \ldots, i_n\}$. That is, Q_I measures the document space similarity when all the terms in set I have been deleted from the index language.

Since deletion of discriminators raises Q and deletion of non-discriminators lowers Q, some optimal set of terms I_{min} should exist such that $Q_{I_{min}}$ is minimal. It still remains to be shown that the index language consisting of the set of keywords remaining when the set I_{min} is deleted from the total collection of keywords will be optimal in the sense of Keen. If the total set of keywords is $K = \{i_1, i_2, \ldots, i_t\}$, and $I_{min} = \{i_1, \ldots, i_{min}\}$, $min \leq t$, then Figure 1 describes what should happen to Q as terms are successively deleted from K (a point (i_j, Q) represents $Q_{\{i_1, \ldots, i_j\}}$, i.e., Q for the index language given by $K - \{i_1, \ldots, i_j\}$). As non-discriminators are deleted, the document space spreads out and Q goes down to its minimum. Then as discriminators are deleted, documents that were distinguished are coalesced, the document space draws together, and Q goes up (until all documents are identically null).

It may be hypothesized that retrieval will follow the same pattern. That is, using some method of retrieval evaluation, the best results will occur at $Q_{i_{min}}$, and as Q increases, retrieval "goodness" will decrease. One measure of retrieval effectiveness is the rank of the last relevant document retrieved. If N_r is the average rank (over a group of queries) of the last relevant document retrieved, then assuming retrieval follows Q, N_r versus i will be as in Figure 2. As non-discriminators are deleted (i_1 to i_{min}), it is easier to find the relevant documents, and N_r goes down until i_{min} is reached. At that point discriminators begin to be lost, the document space closes up, relevant documents move closer to non-relevant,

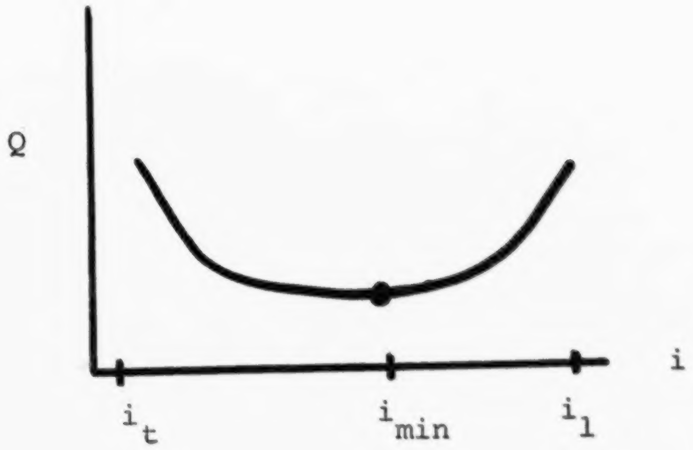


Figure 1

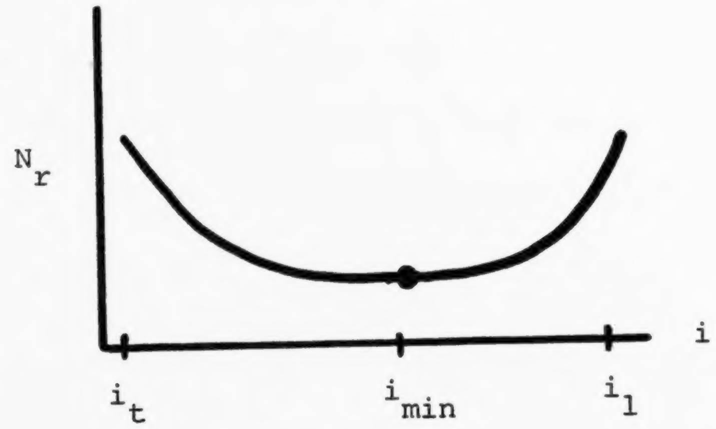


Figure 2

more non-relevant are retrieved along with relevant, and N_r goes back up.

3. Experimental Results

The ADI abstracts collection is used as a base for testing the above predictions about the Q and N_r curves. The full (no common words deleted) vectors and the accompanying word stem dictionary are used. The dictionary terms are ranked twice:

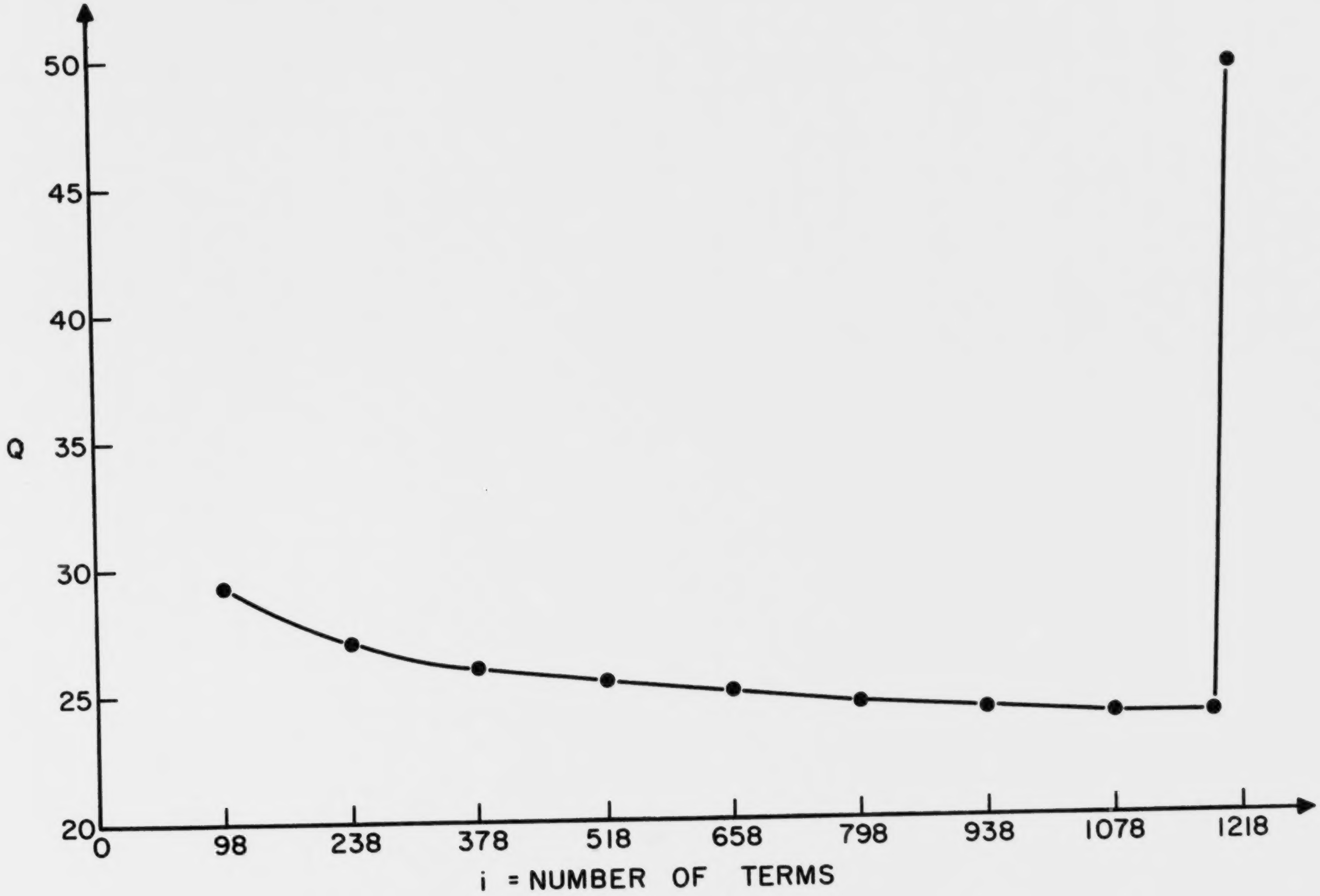
a) in order of increasing Q_i, i.e., with the supposed discriminators at the end of the list;

b) in order of decreasing frequency of occurrence (number of documents appeared in), with the least frequent terms at the end.

Since the ADI collection contains 1218 keywords, only every 28th (an arbitrary number) point of the curves is considered, i.e., what happens when terms 1-28, 1-56, 1-84, . . . are deleted (using the orderings above). At the selected cutoffs, query searches are performed, and the corresponding Q_I's and N_r's calculated.

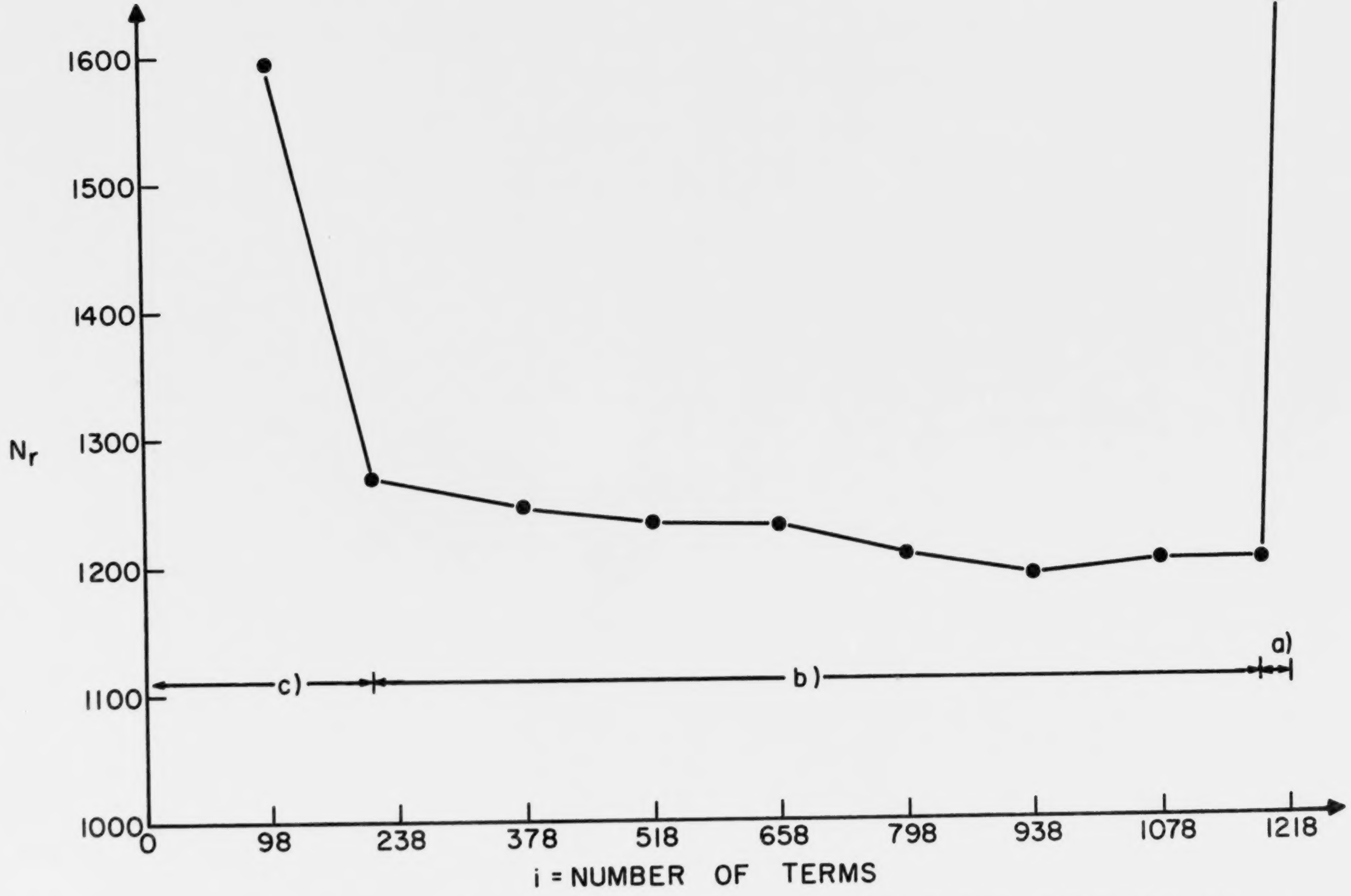
When the terms are deleted in increasing Q_i order, the Q_I and N_r curves come out very much as predicted (Figure 3 and 4), being both of approximately the same shape: dipping down to a minimum and shooting off at both ends (see Figure 5 for comparison). Interestingly, no documents are "lost" (have all their keywords deleted) until all but 98 keywords are deleted, at which time N_r shoots up, indicating that these 98 terms are real discriminators. Also, the N_r curve has a very large, flat middle "minimum" (discounting noise) area — deleting 28 or 36 x 28 terms does not make much difference.

The keywords are thus divided into 3 sets (Figure 4):



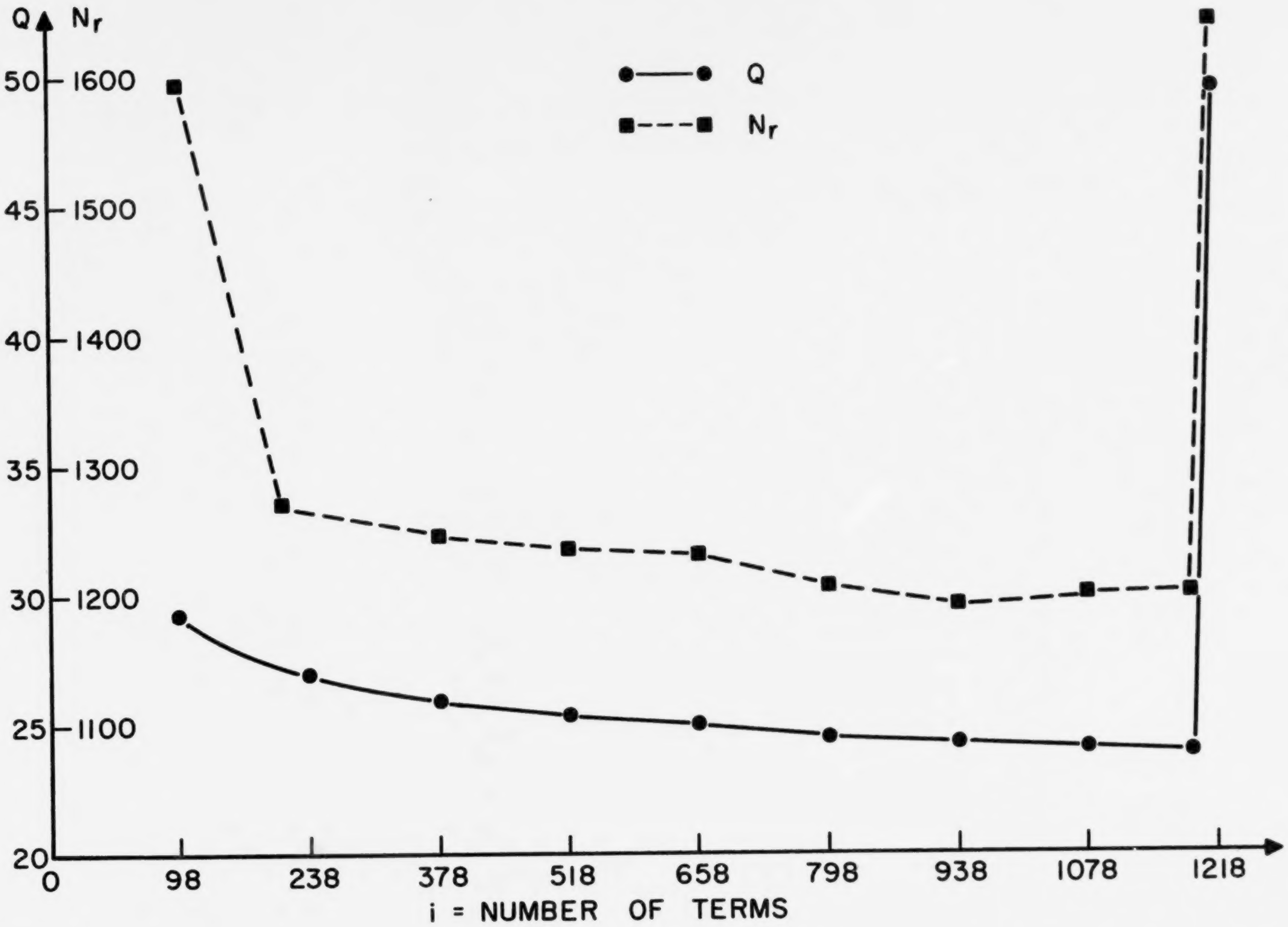
Q vs NUMBER OF CONCEPTS – DELETED BY Q ORDER

Figure 3



N$_r$ (TOTAL FOR 33 QUERIES) vs NUMBER OF CONCEPTS—DELETION BY Q ORDER

Figure 4



Q vs NUMBER OF CONCEPTS

N_r vs NUMBER OF CONCEPTS – DELETION BY Q ORDER

Figure 5

a) those on the right end whose deletion leads to better retrieval (lower N_r);

b) the middle terms which do not make much difference;

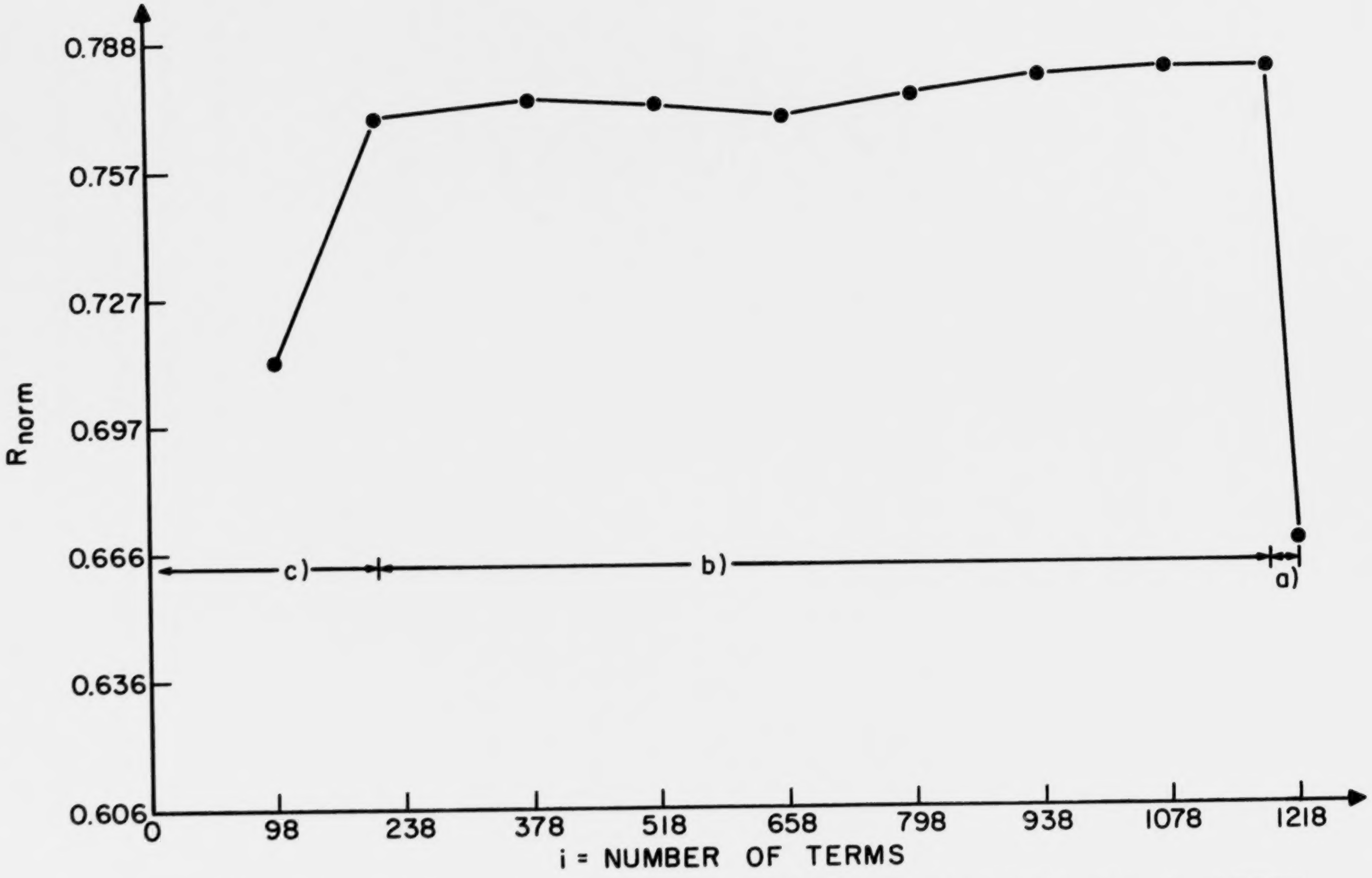
c) those at the left end which must be retained for good retrieval.

The sharp drop on the right-hand side of the curves is somewhat misleading. If all the points along the drop were plotted (corresponding to deleting 1, 2, 3, . . ., 28 keywords), it could be seen that the minimum actually occurs after the first 10 terms are deleted. These 10 terms constitute the set a), and it turns out that for all 10 terms, $Q_i < Q$ (Q without subscript is Q for the full index language). That is, these terms are of the type which according to predictions could be dropped from the index language, and the N_r curve shows that they should be. For all other terms (sets b) and c)), $Q_i > Q$. The members of set a) are therefore easy to identify and include in a negative dictionary: calculate Q for the full index language and Q_i for each keyword and put in the negative dictionary those keywords with $Q_i < Q$.

The normalized recall, defined by

$$R_{norm} = 1 - \frac{\sum_{i=1}^{n} (r_i - i)}{n \cdot (N - n)}$$

for N the total number of documents, n the number of relevant documents and r_i the rank of the i^{th} relevant document retrieved, is an alternate measure of retrieval effectiveness. The curve of normalized recall vs. terms deleted (Figure 6) delineates the same sets a), b), and c) that the N_r curve did. Since high recall is an indication of good retrieval (as opposed to low N_r), inverting the recall curve (by subtracting all values from 1) is required to



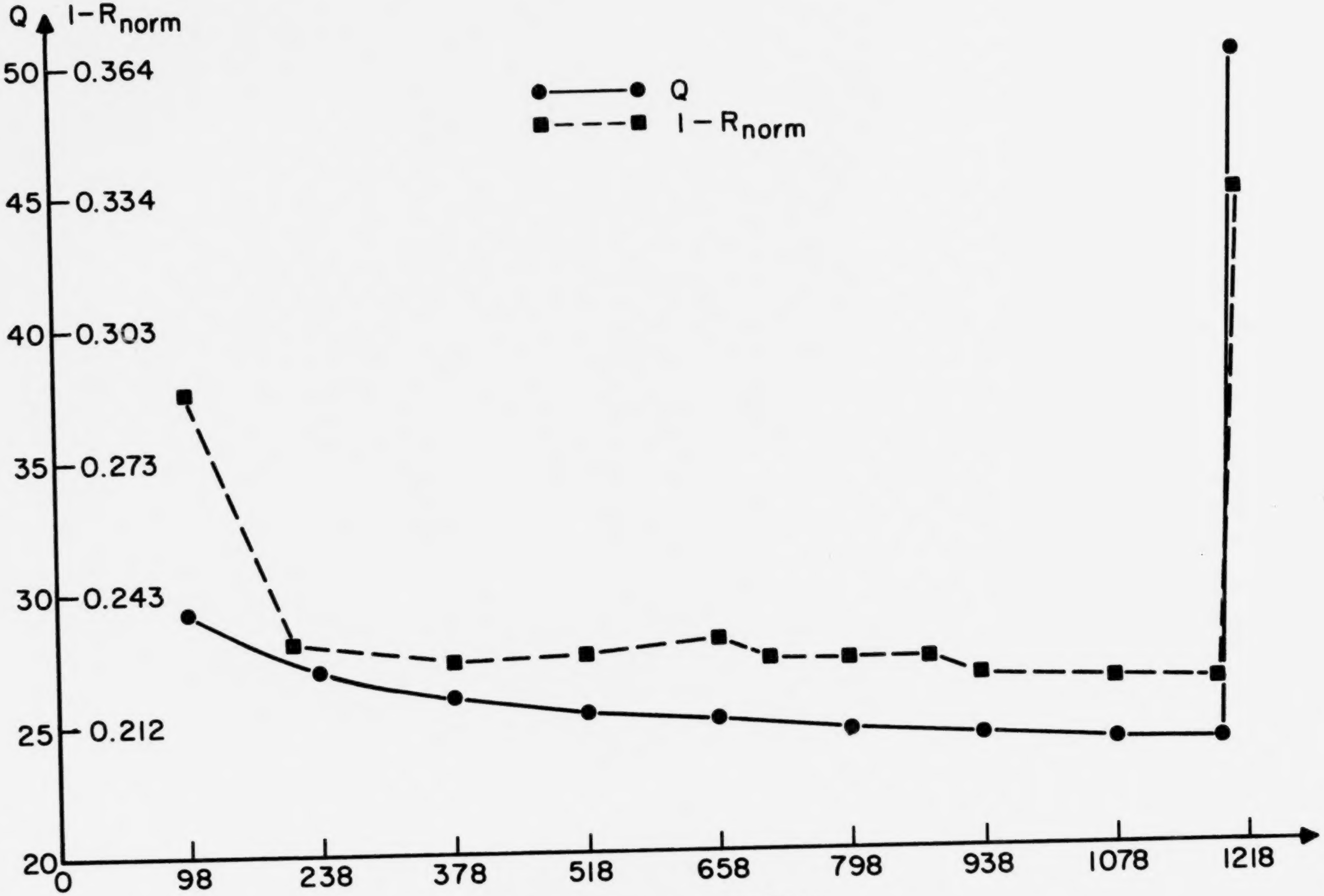
NORMALIZED RECALL vs NUMBER OF CONCEPTS-DELETION BY Q ORDER

Figure 6

show that recall also follows the pattern of Q (Figure 7).

It is interesting to note the frequency classes into which the sets a), b), and c) fall. The non-discriminating members of set a) exhibit the highest frequencies (40% - 100%); the "in-between" members of set b) have the lowest frequencies (0% - 10%), while the discriminators of set c) have 10% - 40%. While the terms in each set occur in the above ranges, within a set they are not exactly in frequency order. Therefore, in terms of frequency, the dividing line between discriminators and non-discriminators is not a clear one, and its absolute value (here, 40%) is likely to change from collection to collection. The use of relative Q's to separate out the non-discriminators, however, does not require the choice of such a cut-off point, and is an easier criterion to apply in constructing a negative dictionary.

When the terms are deleted in decreasing frequency order, the predicted curves do not show up (Figure 8 and 9). Q is strictly decreasing (reading from the right) — the more terms deleted, the more the space spreads out. Since the terms are dropped in approximately the order a), c), b), the loss of non-discriminator a) terms causes the same initial dip. Since the c) terms occur in more documents (have higher frequencies) than the b) terms, deleting them continues the process of spreading out the document space, until documents are identified only by a stray, "rare" word from set b). (In Q order, deleting terms from set b) has the opposite effect; documents that were "pulled away" from the centroid by odd words now move in closer together as terms from set b) are deleted, and Q goes up.) N_r has its initial dip resulting from the loss of the terms of set a), and then rises sharply as the discriminating terms of set c) are lost and the remaining keywords prove to be poor identifiers. In this case, documents

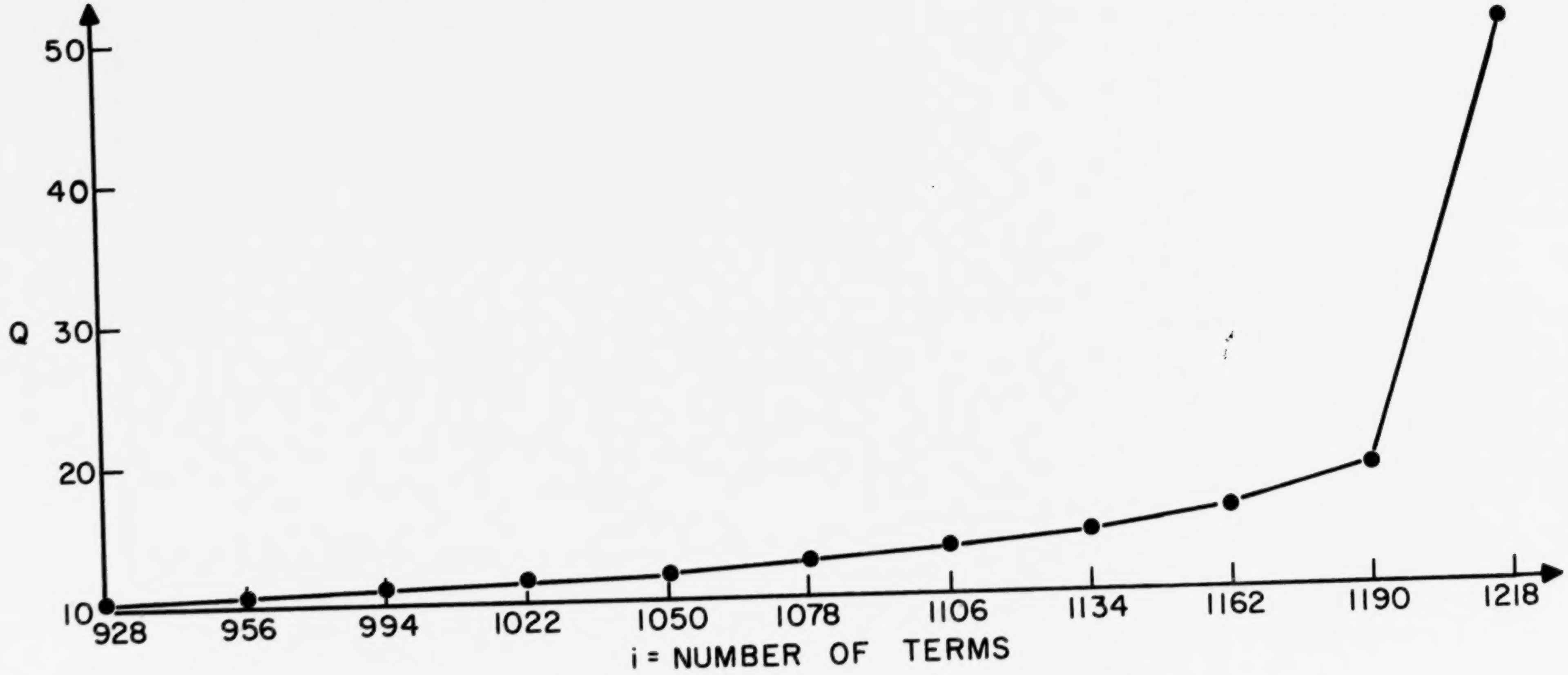


i = NUMBER OF TERMS

Q vs NUMBER OF CONCEPTS

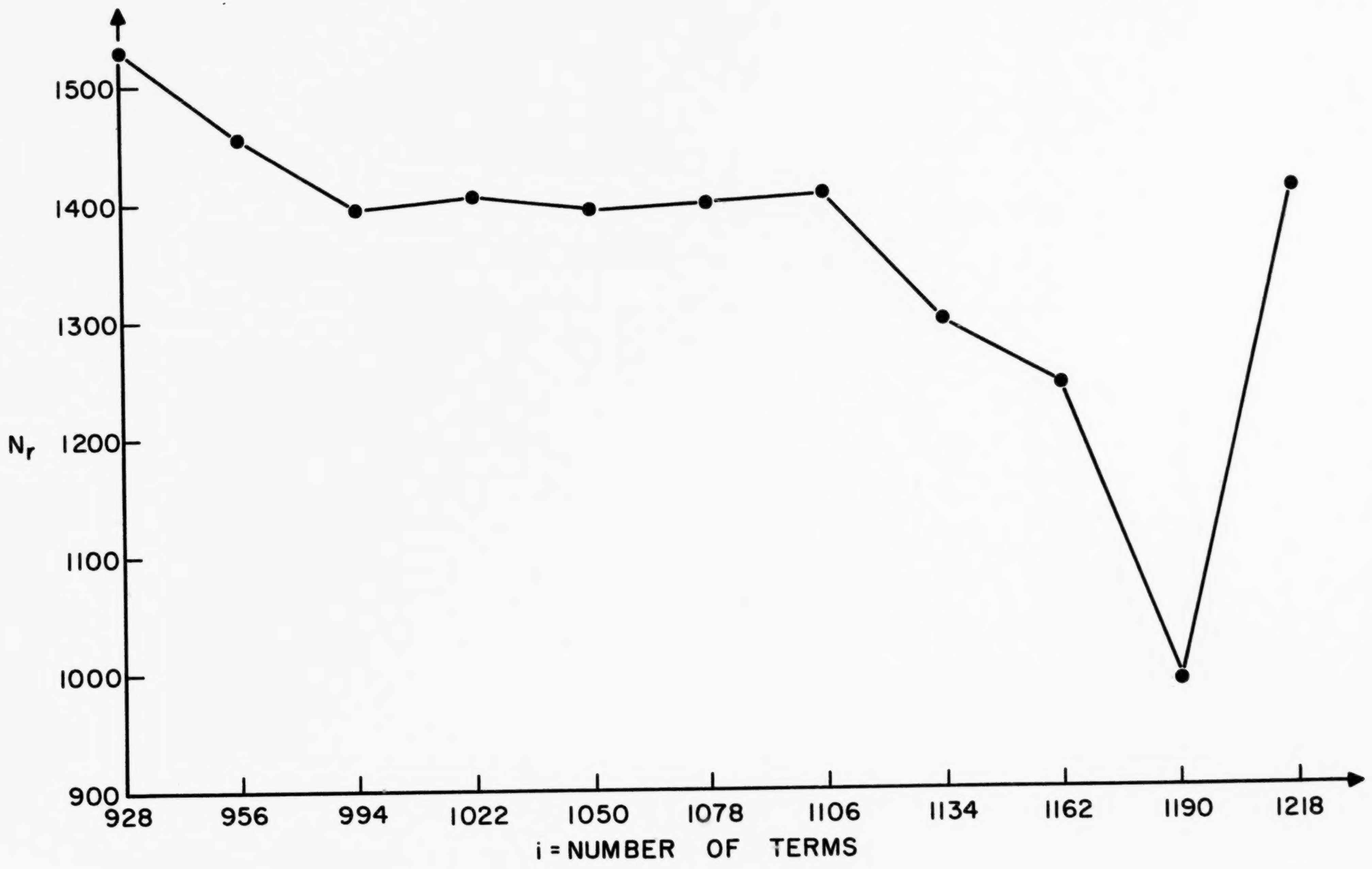
$1-R_{norm}$ vs NUMBER OF CONCEPTS—DELETED BY Q ORDER

Figure 7



Q vs NUMBER OF CONCEPTS - DELETED BY FREQUENCY

Figure 8



N_r (TOTAL FOR 29 QUERIES) vs NUMBER OF CONCEPTS—DELETED BY FREQUENCY

Figure 9

are "lost" much more quickly, after only 560 keywords are deleted.

It is interesting to look at the keywords that fall into sets a), b), and c). Table 1 gives the 10 members of set a) in increasing Q order and their frequencies of occurrence (out of 82).

Keyword	Frequency
off	78
the	77
and	80
a	62
in	61
for	54
to	53
information	44
is	46
are	38

Table 1

Nine of the ten are identifiable as "common function words" without particular semantic content. The tenth, the term "information", also shows up as a non-discriminator, for this particular collection. Since the ADI collection covers documentation, this is not surprising. The fact that "information" does occur in set a) is an indication that the Q criterion will be helpful in constructing negative dictionaries tailored to the collection with which they will be used.

When 40 x 28 terms are deleted, the 98 which remain comprise set c), the so-called discriminators. Many of the 98 can classify as "content words" — "request", "education", "thesaurus", "retrieve" (see Table 2). On the other hand, several "function words" also occur, e.g., "at", "as", "it", "not", "has", "was". That is, in the ADI collection composed of abstracts (rather than full texts), these words serve to "distinguish" between those

Keyword	Frequency	Keyword	Frequency	Keyword	Frequency
index	19	usage	12	tape	7
library	10	procedure	7	produce	11
science	12	national	6	role	8
exchange	3	chemical	5	manual	6
search	12	program	17	recognition	3
process	14	publication	6	editing	2
service	10	journal	10	new	11
documents	19	logic	4	been	13
center	7	reference	6	not	4
definition	3	as	23	rules	2
technical	9	mechanized	3	remote	1
computer	23	it	9	interrogation	1
read	6	communication	7	microfilm	4
character	5	test	5	has	15
copy	7	can	11	prepare	5
be	16	education	4	graduate	3
book	3	material	4	into	5
use	13	by	27	an	27
at	18	concept	7	training	6
retrieve	28	need	11	that	11
analysis	7	level	3	abstract	5
file	6	organization	7	catalogue	1
date	14	facet	1	mathematical	1
thesaurus	4	vocabulary	4	access	5
system	33	have	10	store	7
from	17	or	15	handle	8
method	13	which	14	school	4
page	5	citation	4	literature	5
transfromation	2	comparison	4	word	5
machine	11	relation	5	was	5
image	1	request	5	IBM	4
text	7	foreign	1	name	2
automatic	8	special	8		

Keywords are in decreasing Q_i order, reading down the columns. That That is, "index" is the best discriminator, being better than "technical", which is better than "usage", which is better than "tape", which is better than "name", which is the worst discriminator in set c).

Set c) — Discriminators

Table 2

"documents" in which they appear and those in which they do not. Again, the Q criterion is matching the dictionary to the collection to produce maximal retrieval in a mechanical way without the benefit of human judgment.

The members of set b) appear in an average of two documents each. Both "function words" like "would" and "content words" like "overdue" and "efficiency" are found. Since function words are found in all three sets (and therefore at all frequency ranges), it is clear that a criterion of frequency of occurrence alone is not going to find all function words. At the same time, it will not be a good judge of true discriminators.

4. Experimental Method

The above results are produced in an three-step process:

1) a LOOKUP run produces full document and query vectors, and a list of all word stems used;

2) a FORTRAN program reads document-term vectors, calculates Q_i for each term i and produces a file in increasing Q_i order of keyword concept numbers, frequency of occurrence, and their total sum of weights (over all documents). A second program sorts this file into decreasing frequency order;

3) a third program works with the full documents and query vectors, and either of the term-frequency-weight files to perform the deletion of keywords and the search runs.

A) Calculating Q_i

The first program inverts the document-term vectors and works with this new file and the term-frequency-weight file it creates. It finds the elements of the centroid vector $\underline{c}$ by dividing the total sums of weights for each term by N, the number of documents. To calculate Q, it saves $\sum_{i=1}^{t} v_{ij}^2$ for each document j, and $\sum c_i^2$ for the centroid. Then

$$Q = \sum_{j=1}^{N} \frac{\sum_{i=1}^{t} v_{ij} \cdot c_i}{\sqrt{\sum v_{ij}^2 \cdot \sum c_i^2}} = \frac{1}{\sqrt{\sum c_i^2}} \sum_{j=1}^{N} \frac{\sum_{i=1}^{t} v_{ij} \cdot c_i}{\sqrt{\sum v_{ij}^2}}$$

where t is the total number of terms, and the values of v_{ij} are obtained from the term-document file. As the program goes along, it also saves $\sum_{i=1}^{t} v_{ij} \cdot c_i$ for each document j. Then

$$Q_k = \frac{1}{\sqrt{\sum^{t} (c_i^2) - c_k^2}} \sum_{j=1}^{N} \frac{\sum_{i=1}^{t} (v_{ij} \cdot c_i) - v_{kj} \cdot c_k}{\sqrt{\sum^{t} (v_{ij}^2) - v_{kj}^2}}$$

where the sums to t are all stored values and the values involving k are in the program's files.

B) Deleting and Searching

The third program also inverts the document-term file, and keeps track of $\sum v_{ij}^2$ for all documents j, adjusting the values of the sums as terms are deleted. This program finds $\sum c_i^2$ and calculates $Q_{\{1-28\}}$, $Q_{\{1-56\}}$, . . . , in a manner similar to that described above.

To perform searching a query $\underline{w}$ and its relevancy decisions are read in. Using pointers to keep track of which terms are deleted (which part of the term-document file to ignore), the query is correlated with each document in the collection of full vectors, then with document vectors with 28 terms deleted, then with 56 deleted, and so on. The cosine $\sum v_{ij} \cdot w_i$ /

$\sqrt{\sum v_{ij}^{2} \cdot \sum w_{i}^{2}}$ can be calculated, since the $\sum v_{ij}^{2}$ are stored, the v_{ij} are in the inverted term-document file, and w was just read in. The ranks of the relevant documents can be found by comparing cosines (number of documents with a higher cosine = rank - 1). Typical results are shown in Table 3. The output format is as follows:

the iteration number indicates how many groups of 28 keywords were deleted;

C1 = average cosine of the relevant documents;

C2 = normalized recall;

N_r = rank of last relevant document;

$Q = Q_I$ for the iteration given by the iteration number;

nR ⇛ document n is relevant; the next two numbers are its rank and correlation with the query.

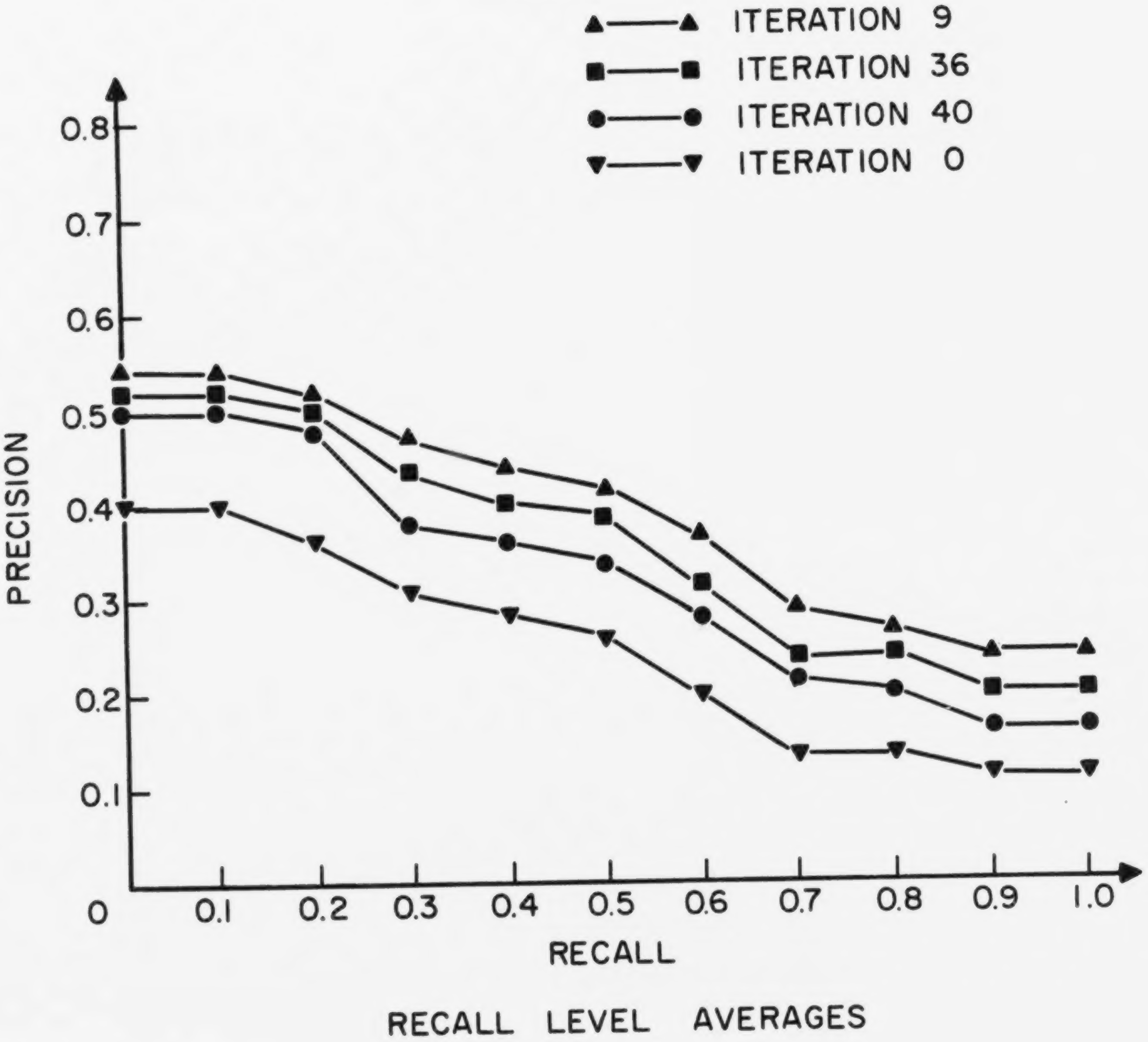
The SMART routine AVERAGE is used to compare retrieval results for different index languages. Some of the results for deleting terms in increasing Q_i order, in particular, iterations 0, 1, 9, 36, and 40, are shown in Figure 10 (which labels these Run 0, 1, 2, 3, and 4, respectively). The recall-precision curves show that deleting concepts does improve retrieval effectiveness. By comparing entries in the table of recall-precision values (Table 4), it can be seen that Run 1 falls on top of Run 2. That is, retrieval performance is about the same whether 28 or 9 x 28 keywords are deleted, but in either case, performance is better than when no terms are deleted. And when only 98 keywords are left (Run 4), the performance is still better than with the full index language (Run 0), falling halfway between best and worst.

To test the effectiveness of the negative dictionary created by the

```
  Iteration 5 Query 24        Iteration 6 Query 24        Iteration 7 Query 24        Iteration 8 Query 24
 Cl=0.196  C2=0.9710807      Cl=0.196  C2=0.9710807      Cl=0.198  C2=0.9710807      Cl=0.199  C2=0.9710807
   NR 22  Q 23.997940          NR 22  Q 24.046610          NR 22  Q 24.113150          NR 22  Q 24.160200

    3R   1  0.3816933          3R   1  0.3816933          3R   1  0.3816933          3R   1  0.3816933
   72R   2  0.2828426         72R   2  0.2828426         72R   2  0.2828426         72R   2  0.2828426
   21R   3  0.2480695         21R   3  0.2480695         21R   3  0.2666666         21R   3  0.2666666
   59R   4  0.2422719         59R   4  0.2422719         59R   4  0.2458614         59R   4  0.2535462
   45R   6  0.1556997         45R   6  0.1556997         45R   6  0.1556997         45R   6  0.1556997
   10R   7  0.1490711         10R   7  0.1490711         10R   7  0.1490711         10R   7  0.1490711
   76R   9  0.1204828         76R   9  0.1204828         76R   9  0.1204828         76R   9  0.1204828
   43R  10  0.1195228         43R  10  0.1195228         43R  10  0.1195228         43R  10  0.1195228
   14R  22  0.0609837         14R  22  0.0609837         14R  22  0.0609837         14R  22  0.0612373


  Iteration 0 Query 25        Iteration 1 Query 25        Iteration 2 Query 25        Iteration 3 Query 25
 Cl=0.426  C2=0.7594937      Cl=0.221  C2=0.8101266      Cl=0.221  C2=0.8101266      Cl=0.221  C2=0.8101266
   NR 54  Q 49.449810          NR 48  Q 23.936350          NR 48  Q 23.936350          NR 48  Q 23.937240

   53R   1  0.5960834         13R   1  0.3608438         13R   1  0.3608438         13R   1  0.3608438
   13R   8  0.4109974         53R   2  0.3015113         53R   2  0.3015113         53R   2  0.3015113
   24R  54  0.2695820         24R  48  0.0000000         24R  48  0.0000000         24R  48  0.0000000
```

Typical Output

Table 3



▲——▲ ITERATION 9
■——■ ITERATION 36
●——● ITERATION 40
▼——▼ ITERATION 0
PRECISION
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
RECALL
RECALL LEVEL AVERAGES

Figure 10

Run 0 — 33 Queries (Plus 0 Nulls) — ADI-82 Full
Run 1 — 33 Queries (Plus 0 Nulls) — ADI-82 Minus 28
Run 2 — 33 Queries (Plus 0 Nulls) — ADI-82 Minus 252
Run 3 — 33 Queries (Plus 0 Nulls) — ADI-82 Minus 1092
Run 4 — 33 Queries (Plus 0 Nulls) — ADI-82 Minus 1120

	Run 0		Run 1		Run 2		Run 3		Run 4	
Recall	NQ	Precision	NQ	Precision	NQ	Precision	NQ	Precision	NQ	Precision
0.0	0	0.4027	0	0.5367	0	0.5498	0	0.5271	0	0.5069
0.05	1	0.3951	1	0.5367	1	0.5405	1	0.5271	1	0.5069
0.10	2	0.3926	2	0.5367	2	0.5405	2	0.5271	2	0.4976
0.15	4	0.3926	4	0.5177	4	0.5215	4	0.5098	4	0.4976
0.20	11	0.3638	11	0.5122	11	0.5159	11	0.4991	11	0.4769
0.25	16	0.3537	16	0.4787	16	0.4819	16	0.4501	16	0.3948
0.30	16	0.3277	16	0.4637	16	0.4623	16	0.4464	16	0.3851
0.35	22	0.2749	22	0.4439	22	0.4454	22	0.4035	22	0.3558
0.40	22	0.2740	22	0.4439	22	0.4454	22	0.4035	22	0.3551
0.45	22	0.2615	22	0.4264	22	0.4280	22	0.3832	22	0.3424
0.50	29	0.2612	29	0.4262	29	0.4277	29	0.3821	29	0.3424
0.55	29	0.2037	29	0.3662	29	0.3669	29	0.3165	29	0.2864
0.60	29	0.2031	29	0.3647	29	0.3658	29	0.3141	29	0.2823
0.65	29	0.2025	29	0.3622	29	0.3603	29	0.3138	29	0.2658
0.70	29	0.1468	29	0.2798	29	0.2748	29	0.2490	29	0.2128
0.75	29	0.1461	29	0.2798	29	0.2748	29	0.2490	29	0.2128
0.80	29	0.1390	29	0.2673	29	0.2671	29	0.2385	29	0.1947
0.85	29	0.1294	29	0.2466	29	0.2529	29	0.2232	29	0.1815
0.90	29	0.1194	29	0.2317	29	0.2391	29	0.2090	29	0.1588
0.95	29	0.1178	29	0.2317	29	0.2391	29	0.2090	29	0.1588
1.00	33	0.1178	33	0.2305	33	0.2379	33	0.2078	33	0.1576
Norm Recall		0.6687		0.7798		0.7789		0.7692		0.7182
Norm Precision		0.4490		0.5750		0.5754		0.5585		0.5084
Rank Recall		0.1459		0.2743		0.2783		0.2498		0.1943
Log Precision		0.2829		0.4043		0.4070		0.3665		0.3387

NQ = Number of Queries used in the average not dependent on any extrapolation.
Norm = Normalized

Recall — Level Averages

Table 4

Q criterion (i.e., the dictionary consists of the terms in set a)), retrieval results should be compared with those obtained on the same collection using the 204 "common English words" list as a negative dictionary. The latter collection is not available on the SMART system, so results are compared with those obtained using the thesaurus dictionary, which lumps synonyms together as well as deleting the 204 words. As shown in Figure 11, the results with the Q negative dictionary (Run 1 = iteration 1) are just about the same as those for the thesaurus, except in the low recall area. Since thesaurus construction involves a large amount of hand work and human judgment while the Q negative dictionary can be generated mechanically, the Q method is preferable if high recall is desired, and the time and effort saved by not preparing a thesaurus may justify the use of the Q method even if precision is the goal.

5. Cost Analysis

The basic rationale for negative dictionaries is that they delete many of the frequent keywords, thus reducing the size of files, and lowering storage and search costs. There is a tradeoff between file size and retrieval effectiveness, and a point of balance between the two has to be found. From Figure 10, it can be deduced that deleting 9 x 28 terms leads to about the same retrieval results as deleting only 28 terms, and if any terms are dropped, all 252 can be. However, deleting 36 x 28 (Run 3) lowers retrieval performance only slightly. Is the saving worth deleting the extra terms?

The question can be rephrased as follows: what is the saving in costs when extra terms from set b) are deleted? The keywords in set a) are deleted to improve retrieval (Figure 10, Run 1). Deletion of keywords in set b) has a lesser effect on retrieval (Run 2 and 3), but the terms in

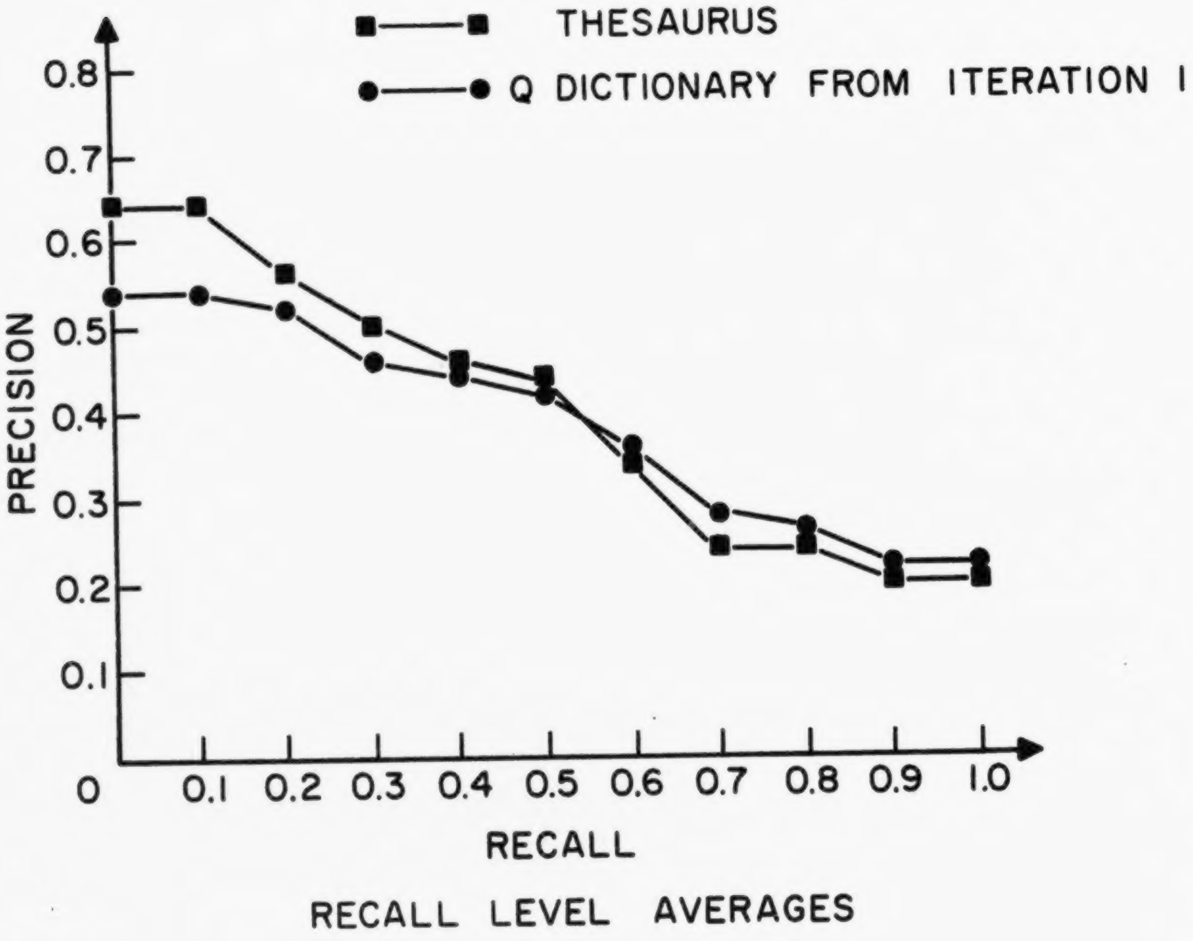

■——■ THESAURUS
●——● Q DICTIONARY FROM ITERATION 1
PRECISION
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
RECALL
RECALL LEVEL AVERAGES

Figure 11

set b) constitute the bulk of the terms to be stored. How much do they cost versus how much do they add to retrieval?

The cost accounting will depend on the system being used and the kind of results it produces. Assume a print-out of all retrieval documents is required and the system works as follows:

a) a full search is performed for each query, processed separately;

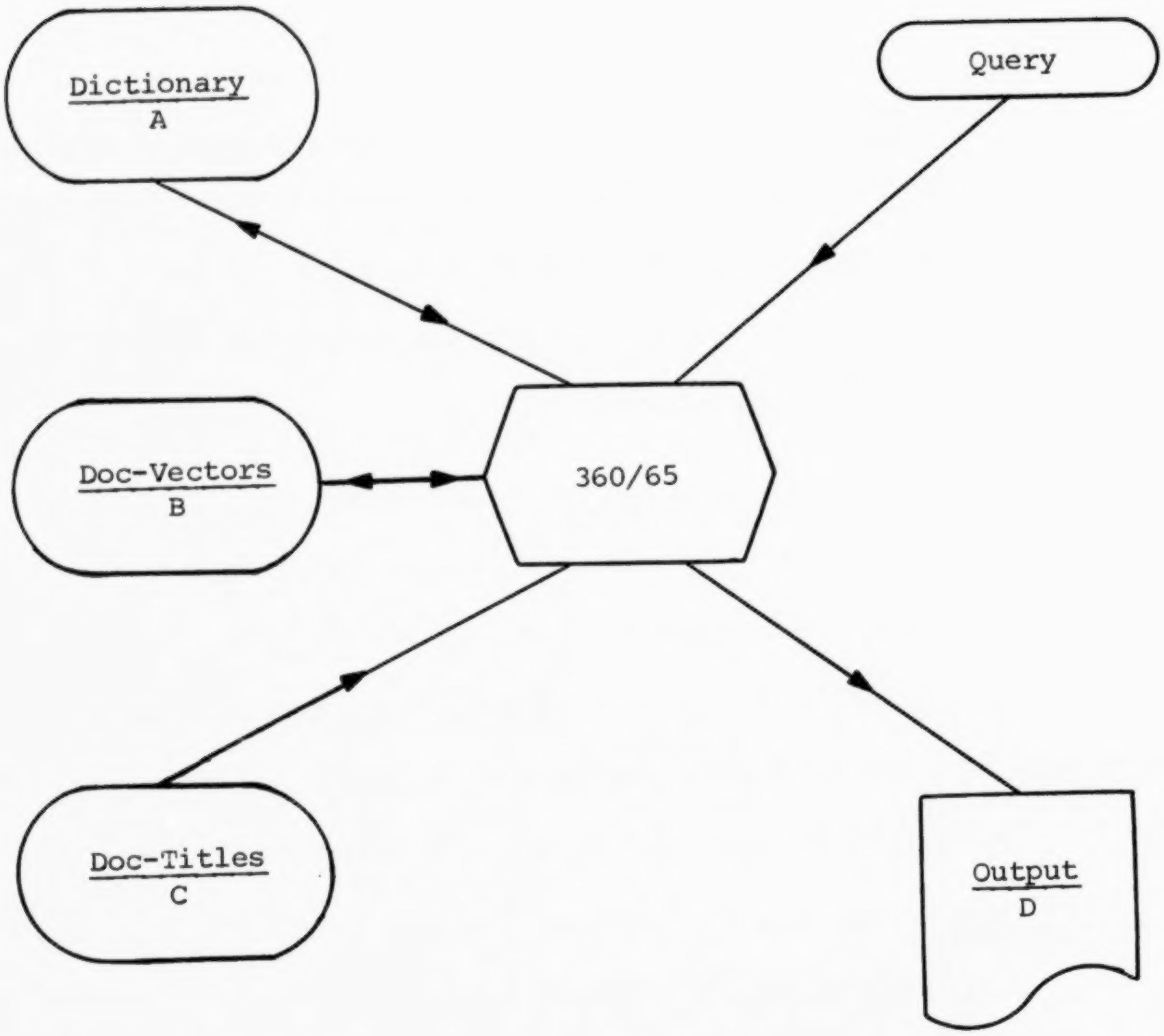
b) results are in the form "Document Title" and "Reference Number", one line per document, with all documents retrieved printed out;

c) the computer is the 360/65 under CLASP;

d) the search program uses 250K and the file organization of the SMART system.

Diagrammatically, the process will appear as in Figure 12. Queries are read in, one at a time, and looked up in the dictionary (A). Each query is correlated with all members of the document file (B) and ranked. The document titles for all documents up to the last relevant are found in the title file (C) and returned to the user (D). (Using all documents up to the last relevant is a convenient measure of how many documents the average user will see.)

What is the dependence of these operations on the total number of terms t? Step (A) is independent of t — each word of the query must be checked for occurrence in the dictionary; non-occurrence takes as long to discover as occurrence. The search step (B) depends on t in two ways: as general file size is reduced, accessing time will go down, and as vector length is reduced, the number of calculations required to compute query-document correlations will be lower. Steps (C) and (D) are independent of t, but are a function of N_r, the rank of the last relevant document


Dictionary
A
Query
Doc-Vectors
B
360/65
Doc-Titles
C
Output
D

System Organization

Figure 12

(since all documents with rank $\leq N_r$ are printed, relevant or not).

Accessing time is related to number of disc tracks read. The ADI collection with all keywords included occupies 4 tracks. Deleting about 200 terms will reduce the number to 3, but even if all the terms found in set b) are deleted, the number of tracks required remains at 3. For 35 queries, the total time saved with reduction to 3 tracks is 1.2 sec. In addition, 50 millisec. is saved in computation time, or for 200 terms deleted, 10 more sec. saved.

The rank of the last relevant document, N_r, generally increases as terms are deleted, resulting in more output lines and an increase in time and cost. Table 5 gives exact figures, in terms of dollars saved, when various numbers of terms are deleted. Figure 13 is a plot of these values, showing the savings in search resulting from deduction from 4 tracks to 3, and the total savings, as functions of the number of terms deleted.

6. Conclusions

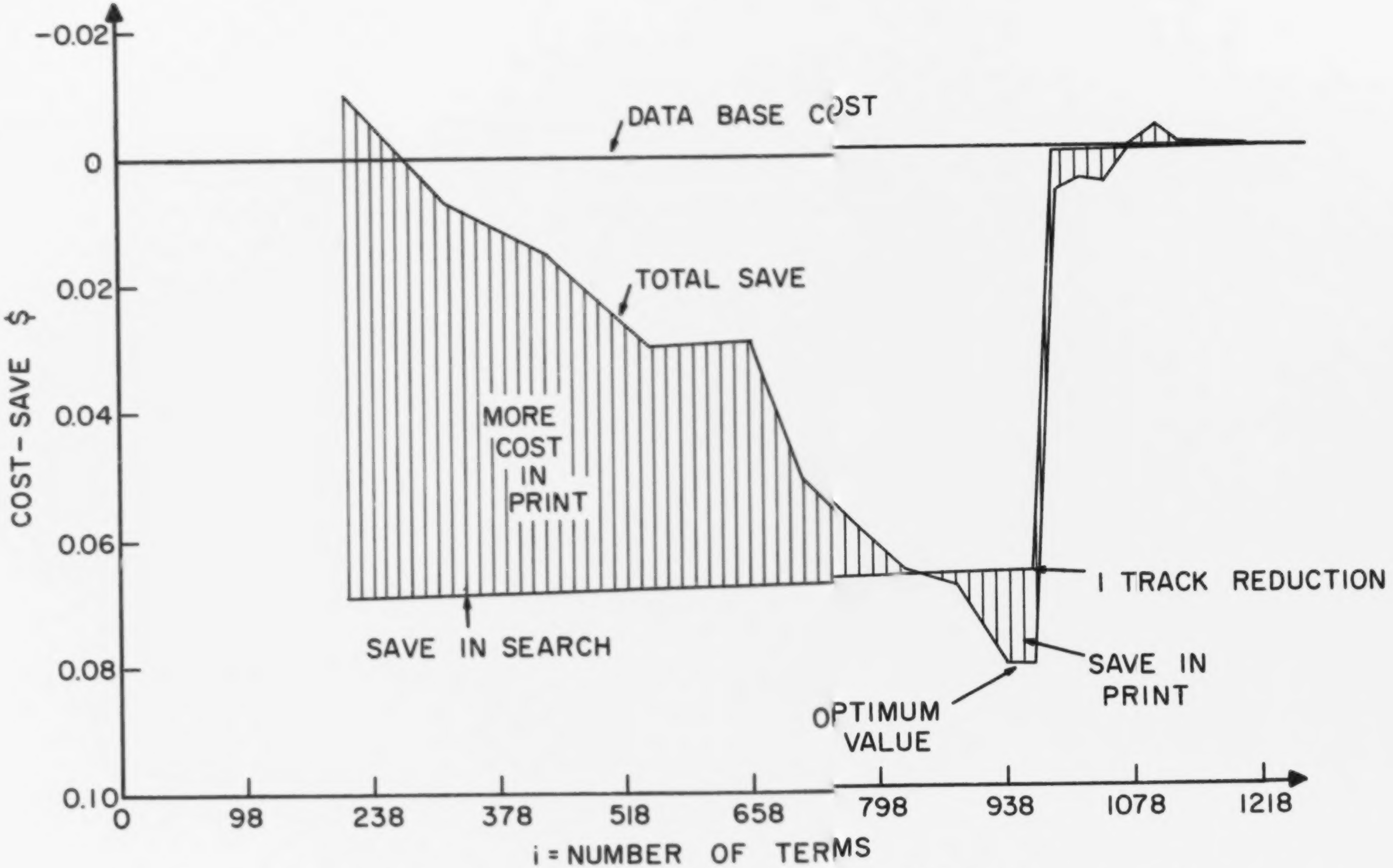
Clearly, a negative dictionary is needed; deletion of some keywords definitely improves retrieval. Deleting words in order of increasing Q seems the better method; while the N_r curve for frequency order has a lower minimum point, it is very unstable. Terms from set a), with $Q_i < Q$, are to be deleted; discriminators from set c) are to be retained. The question of what to do with the middle (set b)) depends on the needs of the user. For a large collection, deleting all but the most vital terms will save storage costs and search time, possibly at some small loss in retrieval. The ADI collection is too small to show very significant differences in cost when terms are deleted.

Number of terms remaining	Number of terms deleted from set b)	Save in Search (dollars)	Decrease in N_r (lines saved)	Save in Print (dollars)	Total Saved (dollars)
1190	0	0.0	0	0.0	0.0
1162	28	0.0	0	0.0	0.0
1134	56	0.00016	0	0.0	0.00016
1106	84	0.00024	- 2	-0.0026	-0.00236
1078	112	0.00033	0	0.0	0.00033
1050	140	0.00042	4	0.0052	0.00562
1022	168	0.0005	3	0.0039	0.0044
994	196	0.0006	5	0.0065	0.0071
966	224	0.0667	11	0.0143	0.0810
938	252	0.0668	11	0.0143	0.0811
882	308	0.0670	1	0.0013	0.0683
826	364	0.0671	- 1	-0.0013	0.0658
770	420	0.0672	- 6	-0.0078	0.0594
714	476	0.0674	-13	-0.0169	0.0505
658	532	0.0676	-29	-0.0377	0.0299
546	644	0.0678	-29	-0.0377	0.0301
434	756	0.0682	-41	-0.0533	0.0149
322	868	0.0685	-47	-0.0611	0.0074
210	980	0.0688	-61	-0.0793	-0.0105

In terms of cost, the optimal number of terms to delete from set b) is about 950.

Cost Statistics

Table 5



COST - SAVE vs NUMBER OF CONCEPTS - REGION "B" DELETED BY Q

Figure 13

The algorithm presented for determing the set a) requires the calculation of Q_i for each term i, and the storage of the entire term-document file. By judicious handling of the values involved, a farily efficient method for discovering set a) is produced. This procedure should be reasonably practical to run on a large collection, at least for generating the initial negative dictionary. Updates for the dictionary when the collection changes could be produced by rerunning the programs on a representative sample of the revised collection.

References

[1] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill Series in Computer Science, 1968.

[2] G. Salton and M. E. Lesk, Information Analysis and Dictionary Construction, Information Storage and Retrieval, Report No. ISR-11 to the National Science Foundation, Section IV, Department of Computer Science, Cornell University, June 1966.

[3] E. M. Keen, Test Environment, Information Storage and Retrieval, Report No. ISR-13 to the National Science Foundation, Section I, Department of Computer Science, Cornell University, December 1967.

VII. Experiments in Automatic Thesaurus Construction for Information Retrieval

G. Salton

Abstract

One of the principal intellectual as well as economic problems in automatic text analysis is the requirement for language analysis tools able to transform variable text inputs into standardized, analyzed formats. Normally, word lists and dictionaries are constructed manually at great expense in time and effort to be used in identifying relationships between words and in distinguishing important "content" words from "common" words to be discarded.

Several new methods for automatic, or semi-automatic, dictionary construction are described, including procedures for the automatic identification of common words, and novel automatic word grouping methods. The resulting dictionaries are evaluated in an information retrieval environment. It appears that in addition to the obvious economic advantages, several of the automatic analysis tools offer improvements in retrieval effectiveness over the standard, manual methods in general use.

1. Manual Dictionary Construction

Most information retrieval and text processing systems include as a principal component a language analysis system designed to determine the "content", or "meaning" of a given information item. In a conventional library system, this analysis may be performed by a human agent, using established classification schedules to determine what content identifiers will best fit a given item. Other "automatic indexing" systems are known in which the content identifiers are generated automatically from document and query texts.

Since the natural language contains irregularities governing both the syntactic and the semantic structures, a content analysis system must normalize the input texts by transforming the variable, possibly ambiguous, input structures into fixed, standardized content identifiers. Such a language normalization process is often based on dictionaries and word lists, which specify the allowable content identifiers, and give for each identifer appropriate definitions to regularize and control its use. In the automatic SMART document retrieval system, the following principal dictionary types are used as an example [1]:

a) a negative dictionary containing "common" terms whose use is proscribed for content analysis purposes;

b) a thesaurus, or synonym dictionary, specifying for each dictionary entry, one or more synonym categories, or concept classes;

c) a phrase dictionary identifying the most frequently used word or concept combinations;

d) a hierarchical arrangement of terms or concepts, similar in structure to a standard library classification schedule.

While well-constructed dictionaries are indispensable for a consistent assignment of content identifiers, or concepts, to information items, the task of building an effective dictionary is always difficult, particularly if the environment within which the dictionary operates is subject to change, or if the given subject area is relatively broad and nonhomogeneous. [2]

The following procedure summarizes the largely manual process normally used by the SMART system for the construction of negative dictionaries and thesauruses [3]:

a) a standard common word list is prepared consisting of function words to be excluded from the dictionary;

b) a keyword-in-context, or concordance listing is generated for a sample document collection in the area under consideration, giving for each word token the context, as well as the total occurrence frequency for each word;

c) the common word list is extended by adding new non-significant words taken from the concordance listing; in general, the words added to form the revised common word list are either very high frequency words providing little discrimination in the subject area under consideration, or very low frequency words which produce few matches between queries and documents;

d) a standard suffix list is prepared, consisting of the principal suffixes applicable to English language material;

e) an automatic suffix removel program is then used to reduce all remaining (noncommon) words to word stem form; the resulting word stem dictionary may be scanned (manually) in order to detect inadequacies in the stemming procedure;

f) the most frequent significant word stems are then selected to serve as "centers" of concept classes in the thesaurus under construction;

g) the word stem dictionary is scanned in alphabetical order, and medium-frequency word stems are either added to existing concept classes, or are used as "centers" of new concept classes;

h) the remaining, mostly low frequency, word stems are inserted as members of existing word classes;

i) the final thesaurus is manually checked for internal consistency, and printed out.

It has been found experimentally that thesauruses resulting from these processing steps operate most satisfactorily if ambiguous terms are entered only into those concept classes which are likely to be of interest in the subject area under consideration — for example, a term like "bat" need not be encoded to represent an animal if the document collection deals with sports and ball games. Furthermore, the scope of the resulting concept classes should be approximately comparable, in the sense that the total frequency of occurrence of the words in a given concept class should be about equal; high frequency terms must therefore remain in classes by themselves, while low frequency terms should be grouped so that total concept frequencies are equalized. [3] A typical thesaurus excerpt is shown in Table 1 in alphabetical, as well as in numerical, order by concept class number. (Class numbers above 32,000 designate "common" words.) [4]

A number of experiments have been carried out with the SMART system in order to compare the effectiveness in a retrieval environment of manually constructed thesauruses, providing synonym recognition, with that of simple word stem matches in which word stems extracted from documents are matched with those extracted from queries. In general, it is found that the thesaurus procedure which assigns content identifiers representing concept classes, rather than word stems, offers an improvement of about ten percent in precision for a given recall level, when the retrieval results are averaged over many search requests.

Alphabetic Order		Numeric Order	
Word or Word Stem	Concept Classes	Concept Class	Words or Word Stems
wide	438	344	obstacle target
will	32032	345	atmosphere meteorolog weather wind
wind	345,233	346	aircraft airplane bomber craft helicopter missile plane
winding	233		
wipe	403		
wire	232,105		
wire-wound	001		

Typical Thesaurus Excerpt

Table 1

A typical recall-precision output is shown in Fig. 1 for thesaurus and word stem analysis processes. For the left-hand graph (Fig. 1 (a)) full document texts were used in the analysis, whereas document abstracts were used to produce Fig. 1 (b).* [5]

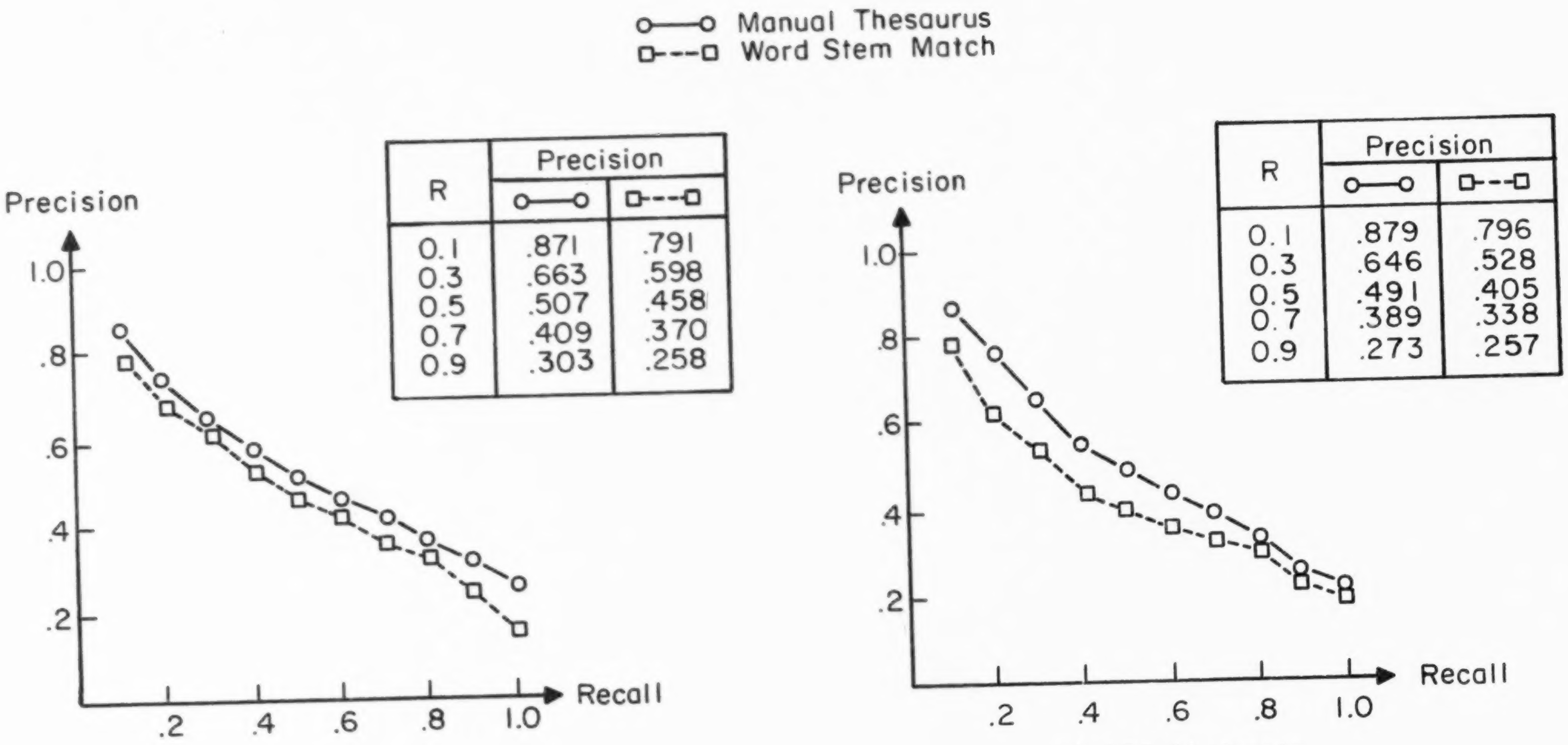
In order to determine what thesaurus properties are particularly desirable from a performance viewpoint, it is of interest to consider briefly the main variables which control the thesaurus generation process [6]:

a) word stem generation
  i) type of suffixing procedure used — whether fully automatic or based on a pre-existing suffix dictionary;
  ii) extent of suffixing — whether based on individual word morphology alone, or also incorporating word context;

b) concept class generation
  i) degree of automation in deriving thesaurus classes;
  ii) average size of thesaurus classes;
  iii) homogeneity in size of thesaurus classes;
  iv) homogeneity in the frequency of occurrence of individual class members (within a thesaurus class);
  v) degree of overlap between thesaurus classes (that is, number of word entries in common between classes);
  vi) semantic closeness between thesaurus classes;

---

*Recall is the proportion of relevant material actually retrieved, while precision is the proportion of retrieved material actually relevant. In general, one would like to retrieve much of what is relevant, while rejecting much of what is extraneous, thereby producing high recall as well as high precision. The curve closest to the upper right-hand corner of a typical recall-precision graph represents the best performance, since recall as well as precision is maximized at that point.



a) ADI Text

R	Precision	
	o—o	□---□
0.1	.871	.791
0.3	.663	.598
0.5	.507	.458
0.7	.409	.370
0.9	.303	.258

b) ADI Abstracts

R	Precision	
	o—o	□---□
0.1	.879	.796
0.3	.646	.528
0.5	.491	.405
0.7	.389	.338
0.9	.273	.257

Comparison of Manual Thesaurus and Word Stem Processes
(Averages over 82 documents, 35 queries)

Fig. 1

c) "common" word recognition
    i) degree of automation in common word recognition process;
    ii) proportion of common words as a percentage of the entire dictionary;
d) processing of linguistic ambiguities
    i) degree of automation in the recognition of linguistic ambiguities;
    ii) extent of recognition of ambiguous structures.

The language analysis procedures incorporated into the SMART document retrieval system all use an automatic word suffixing routine based on a hand-constructed suffix dictionary. Furthermore, linguistic ambiguities represented, for example, by the occurrence of homographs in texts are not explicitly recognized by the SMART analysis process.* The two main variables to be considered in examining thesaurus effectiveness are therefore the common word recognition and the concept grouping procedures. These two problems are treated in the remainder of this study.

2. Common Word Recognition

In discussing the common word problem, it is important, first of all, to distinguish common function words, such as prepositions, conjunctions, or articles, from common content words. The former are easily identified by constructing a list of such terms which may remain constant over many subject areas. The latter, typified by the word stem "automat" in a collection of computer science documents, consist of very high — or very low — frequency terms which should not be incorporated into the standard concept classes of a thesaurus, because the respective terms do not adequately discriminate among the documents in the subject area under consideration. It is important that such words be recognized since their assignment as content identifiers would produce high similarity coefficients between information items which have little in common, and because their presence would magnify the storage and processing costs for the analyzed information items.

---

*Although several language analysis systems use elaborate procedures for the recognition of linguistic ambiguities [7,8], it appears that most potentially ambiguous structures are automatically resolved by restricting the application of a given dictionary to a specific, well-defined subject area.

To determine the importance of the common content word recognition, a study was recently performed comparing the effectiveness in a retrieval environment of a standard word-stem matching process, a standard thesaurus, and a word-stem procedure in which the common content words normally identified as part of the thesaurus process were also recognized. [9] Specifically, a backward procedure was used to generate a word stem dictionary from a thesaurus by breaking down individual thesaurus classes and generating from each distinct word, or word stem, included in one of the thesaurus classes, an entry in the new stem dictionary. The main difference between this new significant stem dictionary and a standard stem dictionary is the absence from the dictionary of word stems corresponding to common functions and common content words normally identified only in a thesaurus. A comparison between significant and standard stem dictionaries will therefore produce evidence concerning the importance of common word deletion from document and query identifications, while the comparison between significant stem and thesaurus dictionaries leads to an evaluation of the concept classes and the term grouping methods used to generate the thesaurus.

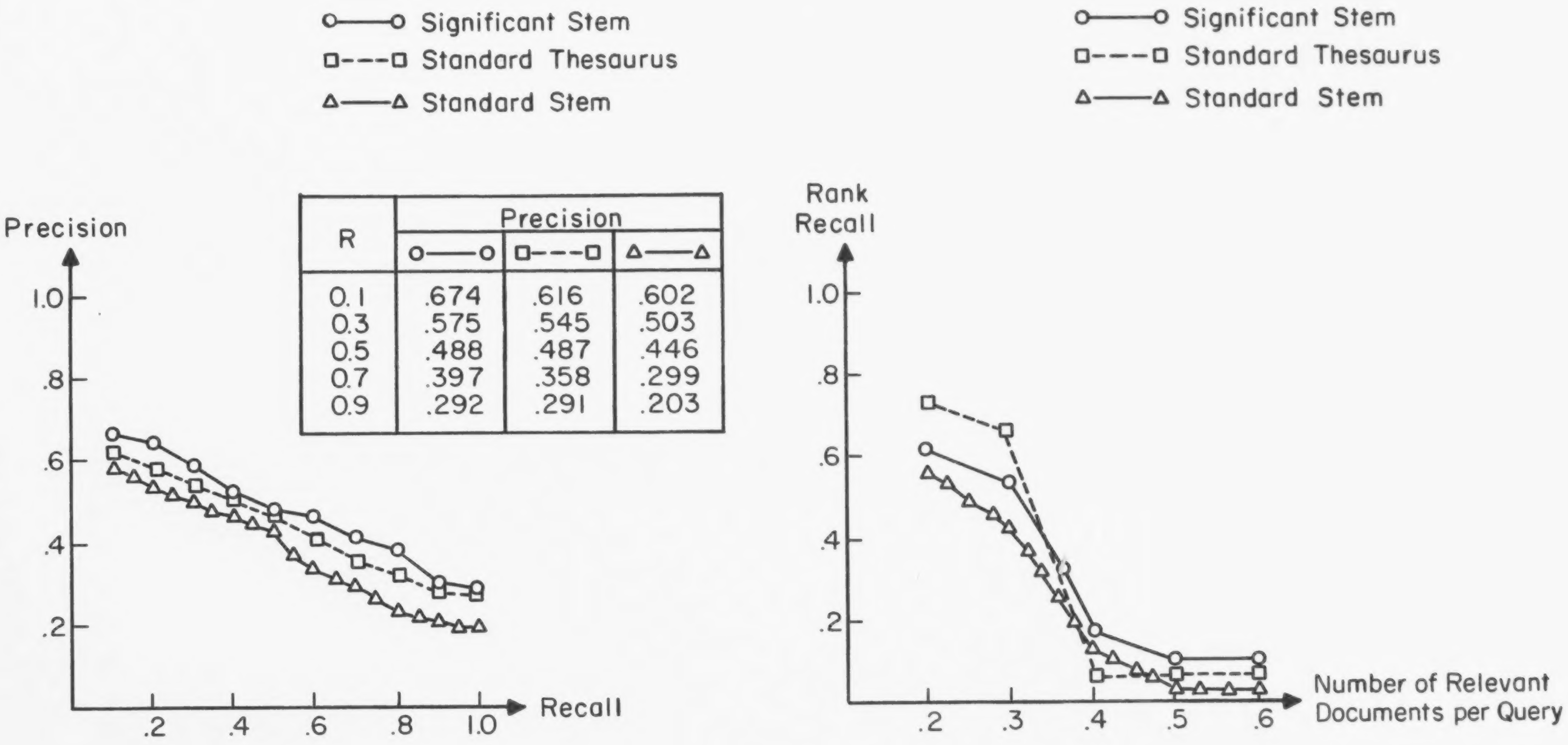
A recall-precision graph for the performance of the three dictionary types is shown in Fig. 2(a), averaged over forty-two queries and two hundred documents in aerodynamics. It may be seen from Fig. 2(a) that the thesaurus produces an improvement of some ten percent in precision for a given recall value over the standard stem process. Unexpectedly, a further improvement is obtained for the significant stem dictionary over the thesaurus performance, indicating that the main virtue of the aerodynamics dictionary being tested is the identification of common words, rather than the grouping of term into concept classes. For the collection under study, the significant stem dictionary contains about twice as many common word entries as the standard stem dictionary.

Obviously, the recall-precision results reflected in the graph of Fig 2(a) cannot be used to conclude that synonym dictionaries, or thesauruses based on term grouping procedures are useless for the analysis of document and query content in information retrieval. Quite often, special requirements may exist for individual queries, such as, for example, an expressed need for very high recall, or precision; in such circumstances, a thesaurus may indeed turn out to be essential.

Consider as an example, the output graph of Fig. 2(b) in which a global evaluation measure, known as rank recall, is plotted for the ten queries (out of forty-two) which were identified by exactly six thesaurus concepts.* It is seen that for queries with very few relevant

---

*The rank recall measure expresses performance by a single number which varies inversely with the ranks achieved by the relevant documents during the retrieval process [1].



R	Precision		
	o——o	□---□	△——△
0.1	.674	.616	.602
0.3	.575	.545	.503
0.5	.488	.487	.446
0.7	.397	.358	.299
0.9	.292	.291	.203

a) Recall–Precision Graph
(200 Documents, 42 Queries)

b) Rank Recall for Queries
with 6 Concepts

Comparison of Significant Stem Dictionary with Thesaurus
and Standard Stem (Cranfield Collection)

Fig. 2

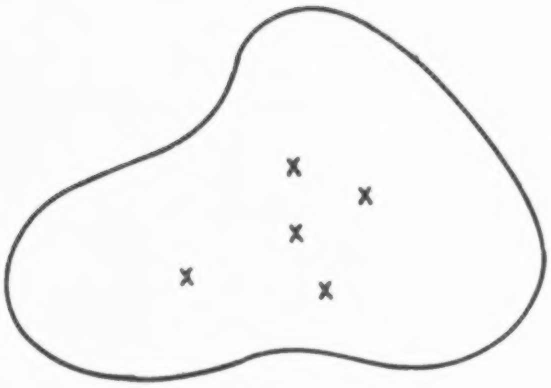
documents in the collection, the thesaurus in fact is able to identify the relevant items more effectively than either of the stem dictionaries. As the number of relevant documents per query increases, the stem methods catch up with the thesaurus process.

In view of the obvious importance of common word identification, one may inquire whether such entries might not be identifiable automatically, instead of being manually generated by the procedure outlined in the previous section. This question was studied using the following mathematical model. Consider the original set of terms, or concepts, used to identify a given query and document collection, and let this term set be altered by selective deletion of certain terms from the query and document identifications. One of two results will then be obtained depending on the type of terms actually removed:

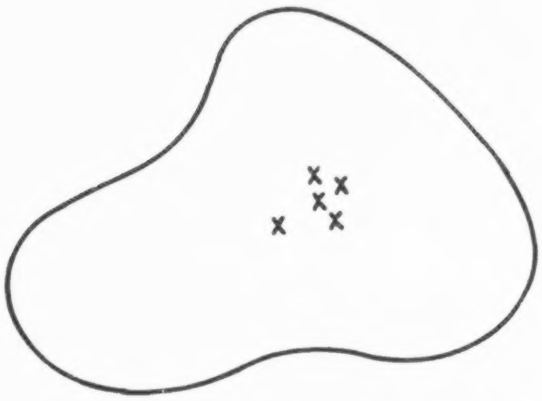
a) if the terms to be removed are useful for content analysis purposes, they will provide discrimination among the documents, and their removal will cause the document space to become more "bunched-up" by rendering all documents more similar to each other, that is, by increasing the correlation between pairs of documents;

b) on the other hand, if the terms being removed are common words which do not provide discrimination, the document space will spread out, and the correlation between document pairs will decrease.

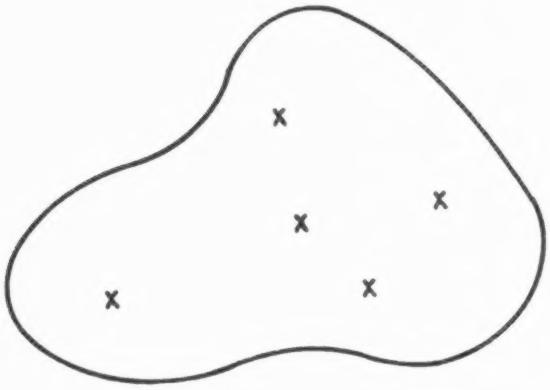
This situation is illustrated by the simplified model of Fig. 3, where each document is identified by 'x', and the similarity between two documents is assumed inversely proportional to the distance between corresponding x's. The conjecture to be tested is then the following: a term



a) Original Document Space



b) Document Space After Removal of Useful Discriminators



c) Document Space After Removal of Useless Nondiscriminators

Changes in Document Space Compactness Following Deletion of Certain Terms

Fig. 3

to be identified as a "common" word, and therefore to be removed from the set of potential content identifiers (and from the set of allowable thesaurus concepts) is one which causes the document space to spread out by decreasing its compactness.

The following procedure is used to verify the conjecture [10]. Consider a set of N documents, and let each document j be represented by a vector of terms, or concepts, $\underline{v}_j$, where $\underline{v}_{ij}$ represents the weight of term i in document j. Let the centroid $\underline{c}$ of all document points in a collection be defined as the "mean document", that is

$$\underline{c}_i = \frac{1}{N} \sum_{j=1}^{N} \underline{v}_{ij} \ ;$$

the centroid is then effectively the center of gravity of the document space. If the similarity, between pairs of documents i and j is given by the correlation $r(\underline{v}_i, \underline{v}_j)$, where r ranges from 1 for perfectly similar items to 0 for completely disjoint pairs, the compactness Q of the document space may be defined as

$$Q = \sum_{j=1}^{N} r(\underline{c}, \underline{v}_j), \qquad 0 \leq Q \leq N$$

that is, as the sum of the similarities between each document and the centroid; greater values of Q indicate greater compactness of the document space.

Consider then the function Q_i defining the compactness of the document space with term i deleted. If $Q_i > Q$, the document space is more compact and term i is a discriminator; contrariwise, if $Q_i < Q$, the space is more spread out, and deletion of term i may produce better retrieval. Since deletion of discriminators raises Q, and deletion of nondiscriminators (common words) lowers Q, an optimal set I of terms must exist such that Q_I becomes minimal.

The following experimental procedure may then be used:

a) consider each term i in order and compute Q_i;

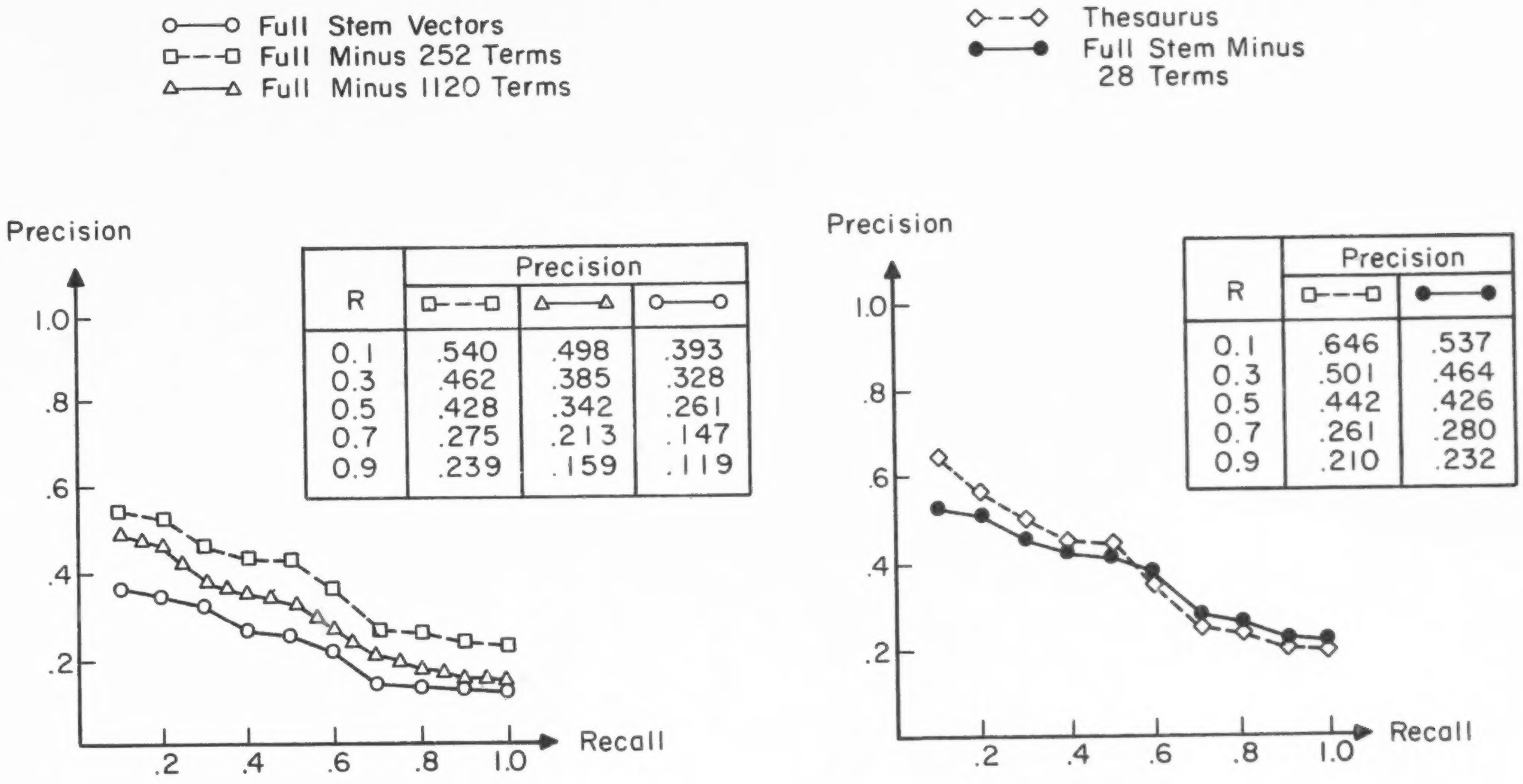
b) arrange the terms in order of decreasing Q_i (that is, with terms causing the greatest decrease coming first);

c) define the set I of common terms to be deleted as the set leading to a minimal Q.

Fig. 4 shows the evaluation results obtained by using this process with a collection of eighty-two documents in the field of documentation, together with thirty-five user queries. A total of 1218 distinct word stems were initially available for the identification of documents. It is seen from Fig. 4(a) that the evaluation results verify the model completely:

a) as high frequency, nondiscriminators are first deleted, the space spreads out, and the corresponding recall-precision output (following deletion of 252 terms) is improved by about twenty percent;

b) when additional terms are deleted, the compactness of the space begins to increase as discriminators are removed, and the recall-precision performance deteriorates; the middle curve of Fig. 4(a) represents the performance following deletion of 1120 terms (in decreasing Q order), at which time the retrieval effectiveness has already diminished by about ten percent.



R	Precision		
	□--□	△—△	○—○
0.1	.540	.498	.393
0.3	.462	.385	.328
0.5	.428	.342	.261
0.7	.275	.213	.147
0.9	.239	.159	.119

a) Comparison of Various Deletion Levels

R	Precision	
	□--□	●—●
0.1	.646	.537
0.3	.501	.464
0.5	.442	.426
0.7	.261	.280
0.9	.210	.232

b) Comparison with Thesaurus

Automatic Common Word Identification
(ADI Collection)

Fig. 4

A comparison between the standard thesaurus performance and a word stem method with the top twenty-eight common terms deleted is shown in Fig. 4(b). Is is seen that the thesaurus process is somewhat superior only at the low recall end with the two graphs being nearly equivalent over most of the performance region.

The results of Fig. 4 thus confirm the earlier studies of Fig. 2 in the sense that word stem matching methods produce performance parameters nearly equivalent to those obtainable by standard thesauruses, providing only that common word stems are appropriately identified, and removed as potential content identifiers.

3. Automatic Concept Grouping Procedures

For many years, the general classification problem consisting of the generation of groups, or classes, of items which are similar, in some sense, to each other has been of major concern in many fields of scientific endeavor. In information retrieval, documents are often classified by grouping them into clusters of items thereby simplifying the information search process. Alternatively, terms or concepts, are grouped into thesaurus classes in such a way that synonyms and other related terms are all identifiable by the same thesaurus class numbers.

In section 1 of this report, various criteria were specified for the manual, or intellectual construction of thesaurus classes. Since the manual generation of thesauruses requires, however, a great deal of time and experience, experiments have been conducted for some years leading to an automatic determination of thesaurus classes based on the properties of the available document collections, that is, on the assignment of terms to documents. The general process may be described as follows [11]:

a) a term-document matrix is first constructed specifying the assignment of terms to documents, including term weights, if any;

b) a term-term similarity matrix is generated from the term-document matrix by computing the similarity between each pair of term vectors, based on joint assignment of terms to documents;

c) a threshold value is applied to the term-term similarity matrix to produce a binary term-term connection matrix in which two terms are assumed connected (that is, a 1 appears in the connection matrix) whenever the similarity between corresponding term vectors is sufficiently high;

d) the binary connection matrix may be viewed as an abstract graph in which each term is represented by a node, and each existing connection as a branch between corresponding pairs of nodes; some function of this graph (for example, the connected components, or maximal complete sub-graphs of the graph) is then used to define the clusters, or classes of terms.*

A number of investigators have constructed term classifications automatically, using procedures similar to the ones outlined above [12, 13, 14]. Unfortunately, the generation of the term-term connection matrix is time-consuming and expensive when the number of terms is not very small. For this reason, less expensive automatic classification methods, in which an existing rough classification is improved by selective modification of the original classes, tend to be used in practice. [15, 16]

---

*A connected component of a graph is a subgraph for which each pair of nodes is connected by a path (a chain of branches); in a maximal complete subgraph, each pair of nodes is connected by a direct branch, and no node not in the subgraph will exhibit such a connection to all other nodes of the subgraph.

To determine the effectiveness of such automatically constructed term classifications in a retrieval environment, three types of experiments are briefly described involving, respectively, an automatic refinement of already existing classes; two fully automatic term classification methods; and a semi-automatic classification process.

The first of these experiments consists in taking an existing term classification, or an existing thesaurus, and in refining the term classes by removing classes which are highly overlapping. [17] One such algorithm tried with the SMART system was based on the following steps (in addition to steps a) through d) already listed):

e) given the existing term classes, a class-class similarity matrix is constructed, using the procedures already outlined for the term-term matrix;

f) a threshold value is applied to the class-class matrix to produce a binary class connection matrix;

g) each maximal complete subgraph defines a new merged concept class;

h) merged classes that are subsets of other larger classes are removed, the remainder constituting the new merged classification.

This procedure was used to refine the documentation thesaurus originally available for the ADI collection, consisting of eighy-two documents and thirty-five search requests. Two "merged" thesauruses were produced as follows:

a) thesaurus 1 with a total of 156 concept classes and approximately 3.9 concepts per class;

b) thesaurus 2 with a total of 289 concept classes, averaging 1.4 concepts per class. [18]

The global normalized recall and precision values, averaged over the thirty-five queries and exhibited in Table 2, show that some improvement in performance is obtainable with the refining process.

The second, more ambitious group of experiments deals with the fully automatic classification procedures outlined at the beginning of this section. In one such study a large variety of graph theoretical definitions was used to define the term classes, including "strings of terms", "stars", "cliques", and "clumps", and various threshold and frequency restrictions were applied to the class generation methods. [19] In general, it is found that some of the automatic classifications operate more effectively than unclassified keywords, particularly if "strong" similarity connections (with a large threshold value) are used, and only nonfrequent terms are permitted to be classified. A comparison of the automatic classifications with manual thesauruses was not attempted in this case.

Another fully automatic term classification experiment was recently concluded, using procedures very similar to the preceding ones, with a large experimental collection of 11,500 document abstracts in computer engineering. [20] A class refining process was implemented in that case, and many different parameter variations were tried. In the end, only modest improvements were obtained over a standard word stem matching process, the author claiming that

Thesaurus Type	Normalized Recall	Normalized Precision
Original Thesaurus	.800	.610
Merged Thesaurus 1	.830	.640
Merged Thesaurus 2	.830	.650

Merged Thesaurus Performance

Table 2

> "in relation to results yielded by our various (automatic) associative strategies, it must be concluded that retrieval by the simple means of comparing keyword stems provides a very good level of performance." [20, p. 61]

The last term classification experiment is based on a semi-automatic method for generating the original term vectors used to produce the term-term similarity matrix. Specifically, a set of properties is manually generated by asking questions about each term, and properly encoding the answers.* For each term, the corresponding property vector is then defined as the set of answers obtained in response to ten or twelve manually generated questions. When all term vectors are available, one of the automatic classification procedures may be used to obtain the actual thesaurus classification. [3, 21]

Such a semi-automatic dictionary was constructed for documents in computer engineering. Its properties are compared with those of a manually constructed thesaurus in the summary of Table 3. It is seen that the semi-automatic thesaurus classes are much less homogeneous — some classes being very large, and some very small — than the corresponding manual classes. Furthermore, fewer common words are identified in the semi-automatic thesaurus.

The retrieval results obtained with the two thesauruses are included in Fig. 5. It is seen that the semi-automatic thesaurus produces a less effective performance than the corresponding manually constructed dictionary

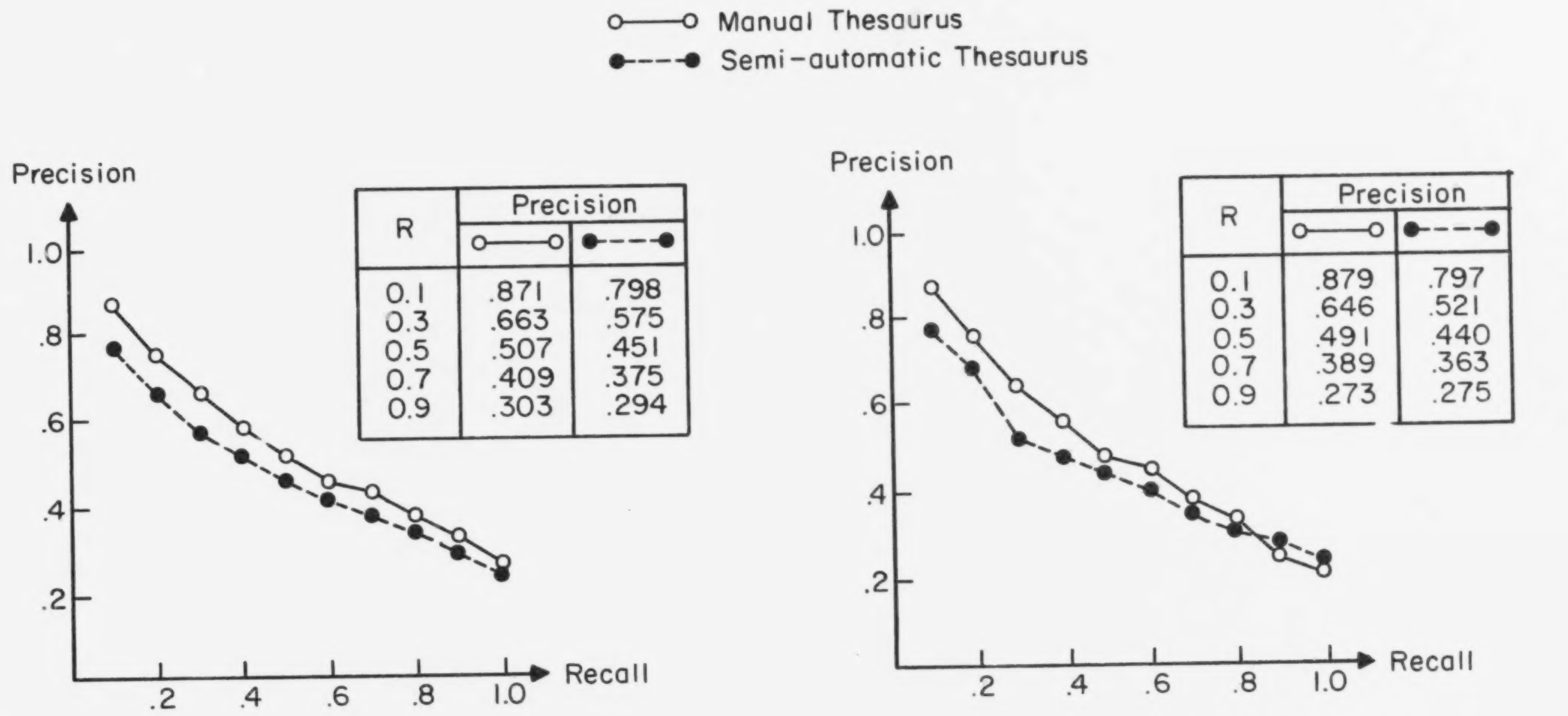
---

*A typical question might inquire whether a given term in computer science refers to computer hardware (1), or to computer software (2), or whether the question is inapplicable to the given term (3); the chosen answer is then encoded by the response number (n).

Properties	Manual (Harris) Thesaurus	Semi-Automatic (Bench) Thesaurus
Number of Concept Classes	863	2953
Number of Word (stem) Entries	2551	5197
Avg. Number of Words per Class	3	1.8
Number of Very Small (Single Word) Classes	468	2725
Number of Very Large Classes (32 to 101 Words)	2	12
Number of Words Appearing in Two or More Classes	52	275
Proportion of "Common" Words Compared to Total Words	37.3%	4.4%

Semi-Automatic Dictionary Properties

Table 3



a) ADI Text

R	Precision ○——○	Precision ●- - -●
0.1	.871	.798
0.3	.663	.575
0.5	.507	.451
0.7	.409	.375
0.9	.303	.294

b) ADI Abstract

R	Precision ○——○	Precision ●- - -●
0.1	.879	.797
0.3	.646	.521
0.5	.491	.440
0.7	.389	.363
0.9	.273	.275

Comparison of Manual and Semi-Automatic Thesaurus

Fig. 5

over most of the performance range. Only for very high recall is the effectiveness of both dictionaries approximately equal.

4. Summary

A number of manual and automatic dictionary construction procedures are described and evaluated in the present study, including in particular, automatic methods for the recognition of common words, and automatic or semi-automatic term grouping methods. It appears that the automatic common word recognition methodology can usefully be incorporated into existing text analysis systems; indeed, the effectiveness of the resulting extended word stem matching process appears equivalent to that obtainable with standard thesauruses.

The effectiveness of the automatic term grouping algorithms is still somewhat in doubt. The automatic grouping methods can probably be implemented more efficiently than the more costly manual thesaurus construction processes. However, no clearly superior automatic thesaurus, using term classes, has as yet been generated. [22, 23]

For the present time, a combination of manual and automatic thesaurus methods therefore appears most promising for practical applications, involving the following steps:

a) automatic common word recognition;

b) manual term classification;

c) automatic refining of the manually produced classes.

References

[1] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill Book Company, New York, 1968.

[2] E. Wall, Vocabulary Building and Control Techniques, American Documentation, Vol. 20, No. 2, April 1969, p. 161-164.

[3] G. Salton and M. E. Lesk, Information Analysis and Dictionary Construction, Scientific Report No. ISR-11 to the National Science Foundation, Section IV, Dept. of Computer Science, Cornell University, June 1966.

[4] M. E. Lesk, Performance of Automatic Information Systems, Information Storage and Retrieval, Vol. 4, 1968, p. 201-218.

[5] G. Salton, Computer Evaluation of Indexing and Text Processing, Journal of the ACM, Vol. 15, No. 1, January 1968, p. 8-36.

[6] A. Moser, Construction of Dictionaries for Text Analysis and Retrieval, unpublished manuscript, Cornell University, 1970.

[7] M. Coyaud and N. Siot-Decauville, L'Analyse Automatique des Documents, Mouton and Co., Paris 1967.

[8] E. B. Fedorov and V. S. Cherniavskii, Automatic Translation from a Natural Language into the Descriptive Language of Systems of the "Empty-Nonempty" Type, in Systems for Handling Information, Moscow, 1969.

[9] D. Bergmark, The Effect of Common Words on Retrieval Performance, Scientific Report No. ISR-18 to the National Science Foundation and to the National Library of Medicine, Dept. of Computer Science, Cornell University, October 1970.

[10] K. Bonwit and J. Aste-Tonsman, Negative Dictionaries, Scientific Report No. ISR-18 to the National Science Foundation and to the National Library of Medicine, Dept. of Computer Science, Cornell University, October 1970.

[11] J. G. Augustson and J. Minker, An Analysis of Some Graph Theoretical Cluster Techniques, Journal of the ACM, Vol. 17, No. 4, October 1970, p. 571-588.

[12] H. Borko, The Construction of an Empirically Based Mathematically Derived Classification System, Report No. SP-588, System Development Corporation, Santa Monica, October 1961.

[13] S. F. Dennis, The Design and Testing of a Fully Automatic Index Searching System for Documents Consisting of Expository Text, in G. Schecter, editor, "Information Retrieval — A Critical View", Thompson Book Co., Washington, D. C., 1967.

[14] K. Sparck Jones and D. Jackson, Current Approaches to Classification and Clump Finding, Computer Journal, Vol. 10, No. 1, May 1967, p. 29-37.

[15] L. B. Doyle, Breaking the Cost Barrier in Automatic Classification Report No. SP-2516, System Development Corp., Santa Monica, July 1966.

[16] R. T. Dattola, A Fast Algorithm for Automatic Classification, Scientific Report No. ISR-16 to the National Science Foundation, Section V, Computer Science Dept., Cornell University, October 1968.

[17] C. C. Gotlieb and S. Kumar, Semantic Clustering of Index Terms, Journal of the ACM, Vol. 15, No. 4, October 1968, p. 493-513.

[18] R. T. Dattola and D. M. Murray, An Experiment in Automatic Thesaurus Construction, Scientific Report No. ISR-13 to the National Science Foundation, Section VIII, Dept. of Computer Science, Cornell University, December 1967.

[19] K. Sparck Jones and D. M. Jackson, The Use of Automatically-Obtained Keyword Classifications for Information Retrieval, Information Storage and Retrieval, Vol. 5, No. 4, February 1970, p. 175-202.

[20] P. K. T. Vaswani and J. B. Cameron, The NPL Experiments in Statistical Word Associations and their Use in Document Indexing and Retrieval, Report Com. Sci. 42, National Physical Laboratory, Teddington, England, April 1970.

[21] G. Salton, Information Dissemination and Automatic Information Systems, Proc. IEEE, Vol. 54, No. 12, December 1966, p. 1663-1678.

[22] C. W. Cleverdon and E. M. Keen, Factors Determining the Performance of Indexing Systems, Aslib-Cranfield Research Project Report, Vol. 1 and 2, Cranfield, England, 1966.

[23] G. Salton, Automatic Text Analysis, Science, Vol. 168, 17 April 1970, p. 335-343.

1. Compactness Function

$$\boxed{Q = \sum_{j=1}^{N} \cos(c, v_j)} \qquad 0 \le Q \le N$$

where

N = total number of documents

c = centroid of document space

v_j = concept vector for document j

2. Term Deletion Algorithm

Let $Q_i = Q$ with term i deleted

If $Q_i > Q$ → document space is more compact
term i is a discriminator

If $Q_i < Q$ → document space is more spread out
term i is a nondiscriminator

Delete set of terms I such that Q_I is minimal

Automatic Common Word Recognition

VIII. Variations on the Query Splitting Technique with Relevance Feedback

T. P. Baker

Abstract

Some experiments in relevance feedback are performed with variations on the technique of query splitting. The results obtained indicate that these variations, as tested, offer no significant improvement over previously tried methods of query splitting.

1. Introduction to Query Splitting

In a document retrieval system with relevance feedback, query splitting refers to the creation of multiple queries from a single previous query, making use of user relevance judgments on documents retrieved by that query in a previous search. The intention in generating these multiple queries is to allow the search to be directed toward several individual clusters of relevant documents, a necessary assumption being that these clusters exist and do contain relevant documents which have not been previously retrieved.

There is little doubt that in a situation where several clusters of relevant documents are retrieved in the initial search it is desirable to generate multiple queries for succeeding iterations. The problem remaining is to distinguish this condition from those in which the relevant documents are unclustered or fall into a single cluster.

Borodin, Kerr, and Lewis [1] propose one method. Their algorithm makes use of the average interdocument correlation among the relevant documents available for feedback as a cutoff in determining whether a given pair of documents should be split. The results obtained with this algorithm are inconclusive, but indicate that it is not sufficiently selective.

Ide [2] suggests that a more sophisticated algorithm might look for separation of relevant documents by nonrelevant documents within the document space, splitting a pair of documents if and only if there exists a nonrelevant document more highly correlated with each of them than they are with each other. In certain respects, this separation criterion is more faithful to the conceptual basis of query splitting than the average correlation criterion. Unlike the average correlation criterion, the separation criterion takes into account the distribution of the nonrelevant documents. This may be significant, since what is desired is the detection of clusters of relevant documents. In contrast, what the average correlation criterion does is to cluster relevant documents. Since nonrelevant documents are not taken into account, this will not produce legitimate clusters, in terms of the whole document space, when relevant documents locally outnumber nonrelevant documents, or vice versa. For this reason it would seem that Ide's untested separation criterion deserves more attention.

The usual concept of query splitting, as discussed by Borodin, Kerr, and Lewis and by Ide, is limited in application to cases where more than one relevant document is retrieved by a previous search iteration. It seems that if query splitting is of any value, something similar could be done for the queries which do not retrieve enough relevant documents to consider splitting in the usual sense. After all, these are generally the queries most in need of modification. What is needed is a dual to the usual formulation of query splitting — a technique of clustering nonrelevant documents for the generation of multiple queries through negative feedback.

2. Algorithms for Query Splitting

Since the algorithm of Borodin, Kerr, and Lewis [1] using the average correlation criterion has been shown to be largely ineffective on the SMART document collections available, and because the separation criterion of Ide [2] remains untried, the primary algorithm tested in this study makes us of the separation criterion.

Since a pair splitting criterion does not by itself define a set of clusters, but rather an association matrix, a splitting algorithm may additionally choose between the use of multilevel associations and the use of direct associations for generating clusters. An examination of the document and query collections used here immediately discloses that multilevel association virtually eliminates cases of splitting in positive feedback. Therefore in order to facilitate experimentation, the splitting algorithm is weakened by permitting only directly connected pairs within clusters used for positive feedback.

Adding to this constraint the requirement that all clusters be maximal, the two conditions are sufficient to define for any pair splitting criterion a unique set of clusters (not necessarily disjoint).

The actual application of these clustering conditions for experimentation with the ADI Abstracts-Thesaurus and Cranfield 200-Thesaurus collections is performed manually using document-document correlations computed by the SMART system. To allow combining the results of the split queries in a consistent fashion, the number of clusters generated for each query (in cases where more would be generated) is limited to two by joining the pair of documents which most nearly fails to pass the separation criterion. The resulting pairs of clusters are fed to the SMART normalized relevance feedback facility in successive iterations.

The SMART relevance feedback formula used is:

$$Q' = MQ + \frac{M}{n} \sum_{i=1}^{n} \frac{R_i}{|R_i|} - \frac{M}{m} \sum_{i=1}^{m} \frac{N_i}{|N_i|} \quad .$$

where Q' is the new query; Q is the original query; M is an integer constant; n is the number of relevant documents (R_i) fed back; m is the number of nonrelevant documents (N_i) fed back.

The top ranking seven documents according to the first "half" of the split query are frozen in place, while the succeeding ranks are determined by another search iteration with the other "half" of the split query. This is done with the two "halves" reversed, as well, so as to average out the effects of order.

The procedure described is applied to all queries retrieving more than one relevant document in the top five ranks according to the first search.

For those queries not retrieving sufficient relevant documents to be split for positive feedback, splitting in negative feedback is attempted.

Where one relevant document is known, the dual to the separation criterion is tried, splitting pairs of nonrelevant documents that are more similar to the one relevant than they are to each other. The resulting clusters of nonrelevant documents are treated like the clusters of relevant documents above, with the single relevant document additionally being fed back with each "half" of the split query.

Where no relevant documents are known, nonrelevant documents are separated by correlation less than the average correlation between documents

Sample separation criterion in its weak form applied to query Q250 of the CRN2TH collection, which retrieved three relevant documents out of five on the first search. The relevant documents retrieved are 3, 115, and 197. The two nonrelevant are 7 and 160.

The interdocument correlation matrix is (in part):

```
3:115 0.5744  3:197 0.4700  3:160 0.4828   3:7  0.2208
            115:197 0.7926 111:160 0.5797 115:7 0.3179
                           197:160 0.5506 197:7 0.3136
```

The pair of relevant documents which must be split for feedback purposes because they are separated by a nonrelevant document is 3:197, which is split by 160.

```
The remaining associations are 3.......115
                                        .
                                        .
                                        .
                                        .
                                        .
                                        .
                                        .
                                       197  ,
```

and the two derived clusters are 3-115 and 115-197.

Separation Criteria for Query Q250

Example 1

Sample separation criterion applied to query B04 of the ADIABTH collection, which retrieves two relevant documents out of the top five ranks according to the first search. The two relevant documents retrieved are 33 and 20. The nonrelevant documents are 5, 46, and 62.

The interdocument correlations are:

33:20	0.1097	33:5	0.4843	33:46	0.2000	33:62	0.2026
		20:5	0.2292	20:46	0.1073	20:62	0.0593

Although it might be interesting to split the nonrelevant documents, there are relevant ones here to split, and the nonrelevant ones are therefore used only to split relevant pairs. We see that the pair 33:20 is split by 5, since 0.4843 and 0.2292 are both greater than 0.1097. Thus 33 and 20 are separated for feedback purposes, and since they are the only relevant documents available they are the two clusters which will be used.

Separation Criterion for Query B04

Example 2

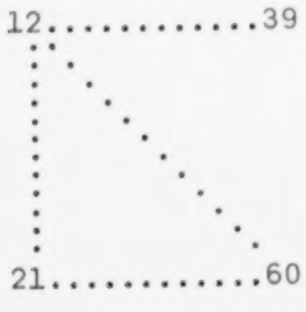
Application of the weak separation criterion to query A13 of the ADIABTH collection, which retrieves one relevant document in the five top-ranked documents according to the first search; the relevant document is 37 and the nonrelevant are 12, 21, 39, and 60.

The interdocument correlation matrix is:

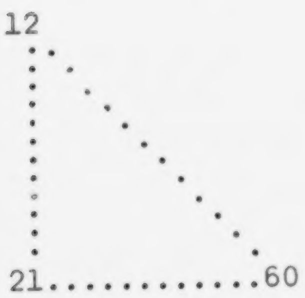
37:12 03411	37:21 0.3659	37:39 0.3225	37:60 0.4000
	12:21 0.3800	12:39 0.3769	12:60 0.3412
		21:39 0.1741	21:60 0.5066
			39:60 0.1061

The following pairs of documents are more highly correlated with 37 than they are with each other, and therefore are separated: 39:60; 21:39.

The remaining associations may be summarized:



Thus the resulting clusters of nonrelevant documents are:



and 12............39 .

Separation Criterion for Query A13
Example 3

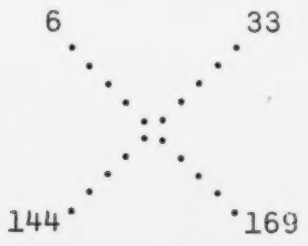
Application of the weak separation criterion to query Q189 of the CRN2TH collection, which retrieves one relevant out of five top-ranking documents according to the first search. The relevant document is 148 and the nonrelevant are 6, 33, 144, and 169.

The interdocument correlation matrix is:

148:6 0.1782	148:33 0.4881	148:144 0.6491	148:169 0.1816
	6:33 0.1630	6:144 0.1347-	6:169 0.2686
		33:144 0.5682	33:169 0.1218-
			144:169 0.0783

The following pairs of documents are more highly correlated with 148 than they are with each other, and therefore are separated for feedback purposes: 144:169; 33:169; 6:144; 6:33.

The remaining associations may be summarized:



The clusters of nonrelevant documents used for feedback are then:

6.........33 and 144..........169 .

Separation Criterion for Query Q189

Example 4

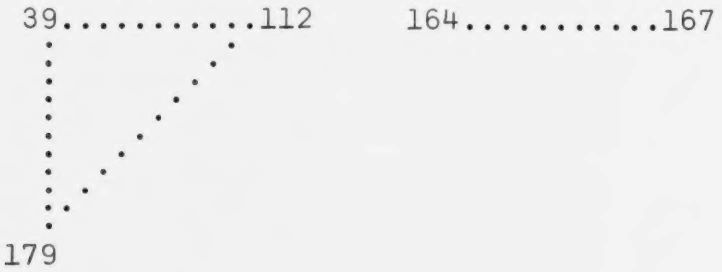
Application of the average correlation criterion to query Q182 of the CRN2TH collection, which retrieved no relevant documents in the top five ranks on the initial search:

Document-document correlations for the nonrelevant documents are:

39:112 .5367 39:164 .0100- 39:167 .0100- 39:179 .5696
112:164 .1142 112:167 .1358 112:179 .6980
164:167 .7212 164:179 .2487
167:179 .1663

The average correlation is 0.3190.

Thus the only associations permitted are:



which are the resulting clusters.

Correlation Criterion for QUERY Q182

Example 5

Application of the average correlation criterion to query Q266 of the CRN2TH collection, which retrieved no relevant documents on the initial search.

The nonrelevant documents known are 58, 162, 163, 164, and 165.

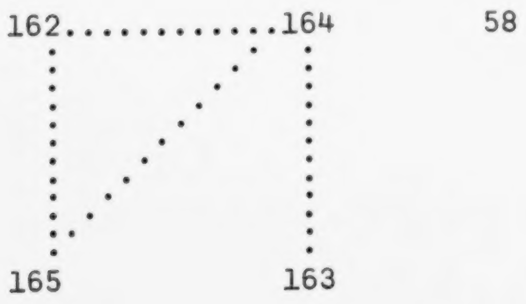
The interdocument correlations are:

58:162 .3932 162:163 .3240 163:164 .5194 164:165 .4585
58:163 .3368 162:164 .5679 163:165 .3744
58:164 .3662 162:165 .5745
58:165 .4113

The average is 0.2819.

Thus the only permissable associations are 162:164, 164:165, 164:163, 162:165.

Thus the clusters are:



Correlation Criterion for Query Q266

Example 6

retrieved, and clusters are formed using multilevel associations (direct associations generally failing to produce any grouping at all). The clusters so formed are fed back in a manner similar to the clusters derived by the other two methods.

Results obtained with these three algorithms on the ADI Abstracts-Thesaurus and Cranfield 200-Thesaurus collections are summarized in the following section.

3. Results of Experimental Runs

The tables on the following pages summarize the results of runs made in the SMART system with splittable queries of the three categories mentioned in the preceding section for the ADI Abstracts-Thesaurus (82 documents and 35 queries — denoted by ADIABTH) and Cranfield 200 (200 documents and 42 queries — denoted by CRN2TH) collections.

The following conventions apply:

-	indicates that the results of the split query and control runs are indistinguishable in terms of the number of relevant documents retrieved.
*	indicates that all relevant documents are retrieved and no improvement is possible.
0	indicates that neither run retrieved any relevant documents.
•	indicates that this query would also have split according to the stronger version of the splitting requirement.
@	indicates a keypunching error detected too late to correct in one of the feedback document specifications for the trial run.

Queries from ADIABTH collection retrieving more than one relevant document on the first search, and splittable by the weak separation criterion:

Query	Improvement of split queries over ordinary normalized positive feedback in terms of relevant documents retrieved up to rank:			
	5	10	15	20
A03/a'	-	-	*	*
A03/b'	-	-	*	*
A15/a'	-1	-	-	-1
A15/b'	-1	-	1	-
B04/a'	-	1	-	-
B04/b'	-1	1	-	-
Average:	-0.5	0.33	0.17	-0.17

Query Splitting Results for ADIABTH Collection
(POSNEG)

Table 1

Queries from the CRN2TH collection retrieving more than one relevant document on the first search, and splittable by the weak separation criterion:

Improvement of split queries over ordinary normalized positive feedback in terms of relevant documents retrieved up to rank:

Query	5	10	15	20
Q122/a˙	-	1	-	-
Q122/b˙	-1	1	-	-
Q148/a˙	-2	-	-	-
Q148/b˙	-2	-	-	-
@ Q250/a	-	-	-	-
@ Q250/b	-	-	-	-
Q268/a	-	-	*	*
Q268/b	-	-	*	*
Q269/a˙	-	-	*	*
Q269/b˙	-	-	*	*
Average:	-4/10	2/10	-	-

Query Splitting Results for CRN2TH Collection
(SPLPOS)

Table 2

Queries from the CRN2TH collection retrieving more than one relevant document on the first search, and splittable by the weak separation criterion:

Improvement of split queries over ordinary normalized positive and negative feedback in terms of relevant documents retrieved up to rank:

Query	5	10	15	20
Q122/a·	-	1	-1	-
Q122/b·	-1	1	-1	-
Q148/a·	-1	-	-	-
Q148/b·	-2	-	-	-
@ Q250/a·	-1	-1	-1	-1
@ Q250/b·	-1	-1	-1	-1
Q268/a	-	-	*	*
Q268/b	-	-	*	*
Q269/a·	-1	-	*	*
Q269/b·	-1	-	*	*
Average:	-8/10	0	-4/10	-1/10

Note: This comparison is unfair to the split query run, since it made no use of negative feedback information.

Query Splitting Results for CRN2TH Collection
(SPLPOS)

Table 3

Queries from the CRN2TH collection retrieving more than one relevant document out of five retrieved on the first search, and splittable by the weak separation criterion:

Improvement of split queries over ordinary normalized positive and negative feedback in terms of relevant documents retrieved up to rank:

Query	5	10	15	20
Q122/a·	-	-	-2	-1
Q122/b·	-	1	-1	-1
Q148/a·	-	-	-	-
Q148/b·	-	-	-	-
@ Q250/a·	-	-1	-1	-1
@ Q250/b·	-	-1	-1	-1
Q268/a	-	-	*	*
Q268/b·	-	-	*	*
Q269/a·	-	-	*	*
Q269/b	-	-	*	*
Average	-	-1/10	-5/10	-4/10

Note: Unlike the other tests, this run was done with the first five documents retrieved by the initial search frozen in their rank positions. It is also unfair to the split query run, since the control made use of negative feedback.

Query Splitting Results for CRN2TH Collection
(SPLPOS)

Table 4

Queries from ADIABTH collection retrieving no relevant documents on the first search, and splittable by correlation less than average:

Query	Improvement of split queries over ordinary normalized negative feedback in relevant documents retrieved up to rank: 5	10	15	20
A08/a	0	-	*	*
A08/b	-1	-	*	*
A09/a	0	-1	-	-
A09/b	1	-	-	-
B11/a	0	0	-	-
B11/b	0	0	-1	-
B13/a	0	-	1	-
B13/b	0	-	-	-
B15/a	0	0	-1	-3
B15/b	0	0	-1	-2
Average:	0	-1/10	-1/5	-1/2

Query Splitting Results for ADIABTH Collection
(ALLNEG)

Table 5

Queries from the CRN2TH collection retrieving no relevant documents in the top five ranks for the first search, and splittable by correlation below average.

Improvement of split queries over ordinary normalized negative feedback with only the top-ranked nonrelevant document used (as opposed to the previous run which used all five nonrelevant available) in terms of relevant documents retrieved up to rank:

Query	5	10	15	20
Q079/a	-1	-1	-1	-
Q079/b	-1	-1	-1	-
Q126/a	0	1	*	*
Q126/a	0	1	*	*
Q132/a	-	-	1	-
Q132/b	-1	-	-	-1
Q182/a	-	-	-	-
Q182/b	-1	-	-	--
Q266/a	0	1	3	2 2
Q266/b	0	2	4	3 3
Q323/a	-	-	-	- -
Q323/b	-	-	-	-
Average:	-4/12	3/12	6/12	4/12

Query Splitting Results for CRN2TH Collection
(NORELS)

Table 6

Queries from the CRN2TH collection retrieving no relevant documents in the first five ranks for the first search, and splittable by correlation below average.

Query	Improvement of split queries over ordinary normalized negative feedback in terms of relevant documents retrieved up to rank: 5	10	15	20
Q079/a	0	0	0	-
Q079/b	0	0	0	-
Q126/a	0	-	*	*
Q126/b	0	-	*	*
Q132/a	-	-1	-	-
Q132/b	-1	-1	-1	-1
Q182/a	-	-	-	-
Q182/b	-1	-	-	-
Q266/a	0	1	-	-
Q266/b	0	2	1	1
Q323/a	-	-	-	-
Q323/b	-	-	-	-
Average:	-2/12	1/12	0	0

Query Splitting Results for CRN2TH Collection
(NORELS)

Table 7

Queries from the ADIABTH collection retrieving one relevant document in the top-ranking five on the first search, and splittable by weak separation criterion for nonrelevant documents.

Improvement of split queries over ordinary normalized positive and negative feedback in terms of relevant documents retrieved up to rank:

Query	5	10	15	20
@ A01·	-1	-	-	-
	-	-	-	-
@ A02·	-	1	*	*
	-	1	*	*
A04·	-	-	-	-
	-	-	-	-
A06·	-	-	-	-
	-	-	-	-
A07	-	-	-	-2
	-	-	-	-
A10	-	-	-1	-
	-	-	-	1
A11	-	-	-	-
	-	-	-1	-
A12·	-	-	-	-1
	-	-	-	-2
A13	-	-	-	-
	-	-	-	-
A14·	-	-	-	-
	-	-	-	-
A17	*	*	*	*
	-1	*	*	*
B16·	-1	-	-	-
	-	-	-1	1
Average:	-3/24	2/24	-3/24	-2/24

Query Splitting Results for ADIABTH Collection
(SPLNEG)

Table 8

Queries from the CRN2TH collection retrieving only one relevant document in the top five ranks on the first search, and splittable by the weak separation criterion for nonrelevant documents.

Query	Improvement of split queries over ordinary normalized positive and negative feedback in terms of relevant documents retrieved up to rank: 5	10	15	20
Q123/a	-	-	-	-
Q123/b	-	-	-	-
Q130/a	-	-1	*	*
Q130/b	-	-1	-1	-1
Q141/a	*	*	*	*
W141/b	*	*	*	*
Q170/a	-	-	-	-
Q170/b	-	-	-	-
Q189/a	*	*	*	*
Q189/b	*	*	*	*
Q272/a	-1	-	*	*
Q272/b	-	-	-1	*
Average:	-1/12	-2/12	-2/12	-1/12

Query Splitting Results for CRN2TH Collection
(ONEREL)

Table 9

Queries from the CRN2TH collection retrieving only one relevant document in the top five ranks on the first search, and splittable by the weak separation criterion for nonrelevant documents.

Query	Improvement of split queries over ordinary normalized positive feedback in terms of relevant documents retrieved up to rank: 5	10	15	20
Q123/a	-	-	-	-
Q123/b	-	-	-	-
Q130/a	-	-	1	-
Q130/b	-	-	-	-1
Q141/a	*	*	*	*
Q141/b	*	*	*	*
Q170/a	-	-	-	-
Q170/b	-	-	-	-
Q189/a	*	*	*	*
Q189/b	*	*	*	*
Q272/a	-	-1	*	*
Q272/b	1	-1	-1	*
Average:	1/12	-2/12	0	-1/12

Query Splitting Results for CRN2TH Collection
(ONEREL)

Table 10

Correlations 0	Correlations 1	Correlations 2	Correlations 3
0.4523	0.6176	0.8241	0.2811
0.3780	0.5598	0.4165	0.1603
0.3665	0.3343	0.4137	0.3070
0.3647	0.3325	0.3680	0.5045
0.3638	0.2731	0.3518	0.4064
0.3467	0.2363	0.3412	0.8449
0.3333	0.2347	0.3246	0.1727
0.3283	0.2334	0.2994	0.4017
0.3119	0.2206	0.2994	0.3635
0.3086	0.2141	0.2946	0.3530
0.3000	0.2130	0.2930	0.3529
0.3000	0.2092	0.2908	0.3390
0.2949	0.2033	0.2768	0.3360
0.2949	0.2001	0.2758	0.3356
0.2917	0.1763	0.2668	0.3350
	0.1606		
	0.1547		
0.2673			
	0.1418	0.2488	
		0.2482	
		0.2415	
0.2080			
0.1803			0.2109
0.1793			
			0.1705
			0.1607

Rank	Documents 0	Documents 1	Documents 2	Documents 3
1	33R	20R	20R	20R
2	5	33R	57R	57R
3	62	42	46	46
4	46	7	5	5
5	20R	9	56	56
6	70	67	33R	33R
7	10	56	7	7
8	11	27	37	10
9	1	5	60	21
10	80	71	21	76
11	37	28	9	37
12	60	18	45	3
13	47	23	12	62
14	56	68	10	60
15	3	24	79	53
17		36R		
19		16R		
20	57R			
22		57R	36R	
23			26R	
26			16R	
29	26R			
32	16R			36R
34	36R			
39				16R
43				26R

	Doc. Corr	Cent. Corr	Drop Doc	Corr. Rank	Old. Doc	Old Reldoc	New Doc
Run 0	82	0	17	65	0	0	0
Run 1	82	0	31	51	0	0	5
Run 2	82	0	12	70	5	2	0
Run 3	75	0	14	61	5	2	7

0 initial search
1 control run with positive and negative feedback
2 first "half" of split query
3 second "half" of split query

Sample Output for Query B04

Fig. 1

4. Evaluation

In general, the results for query splitting in positive feedback with the separation criterion are comparable to those achieved by Borodin, Kerr and Lewis in their experiments with the average correlation criterion. Although a slight improvement may be noted for split queries over ordinary feedback, it is not predictable enough to justify the use of query splitting in a working retrieval system.

Only if a more selective method can be devised for determining which queries will benefit from splitting will the technique become of practical value. Merely strengthening the splitting requirement by permitting multi-level associations in cluster formation appears to be of some value in eliminating nonproductive splitting in the queries tested. All queries split by the weaker method which show an improvement under splitting would be split in like manner by the stronger method. Strengthening the separation as well, by providing that pairs be separated only if they exceed the requirements by some margin, may also be of value in restricting the number of undesired splits.

For negative feedback, the situation is worse. The only run in which splitting exhibited any improvement over the usual negative feedback was on queries in the Cranfield collection retrieving no relevant documents in the first search. Even there, the improvement was erratic. This failure of splitting applied to negative feedback is not entirely surprising, since the hypothesis of separate clusters of relevant documents used to justify splitting in positive feedback does not apply. Here the best justification for splitting is that, since the locations of no relevant documents are known, multiple queries may offer more chance of success by means of a "shotgun" effect — scattering the search over a larger area of the document space.

Altogether, the results of the negative feedback runs indicate that the different "halves" of the split queries do not usually retrieve significantly different portions of the document space. Thus it would seem that this "shotgun" effect is not taking place. It may be that results can be improved by weighting the nonrelevant documents more heavily in feedback. In any case, negative query splitting as tested does not appear to benefit from this effect sufficiently to justify the effort of multiple query generation.*

Although the results of these experiments are largely negative, it is important in viewing them to consider that the queries tested were written by experts in their fields and are therefore generally consistent, thus making the probability of success in query splitting rather low. Also, being small, the document collections used are inimical to the existence of multiple clusters of relevant documents. Relevant documents in such small collections tend to fall into single clusters, or none. Although the success of query splitting in these adverse circumstances would be a strong argument in its favor, its failure in the same circumstances is less conclusive. It would appear that if truly significant results are to be achieved with query splitting, they will be achieved in the environment of a larger more diverse document collection and with more realistically inconsistent queries.

---

* The only exception to this is query Q266 of CRN2TH, which showed remarkable improvement on splitting.

References

[1] A. Borodin, L. Kerr and F. Lewis, Query Splitting in Relevance Feedback Systems; Report ISR-14 to the National Science Foundation, Section XII, October 1968.

[2] E. Ide, Relevance Feedback In An Automatic Document Retrieval System; Report ISR-15 to the National Science Foundation, January 1968.

[3] D. Williamson, R. Williamson, M. Lesk, The Cornell Implementation of the SMART System; Report ISR-16 to the National Science Foundation, Section I, September 1969.

IX. Effectiveness of Feedback Strategies on Collections of Differing Generality

B. Capps and M. Yin

Abstract

This study evaluates the comparative effectiveness of several feedback strategies on collections which differ in generality, namely the Cranfield 200 and Cranfield 400 collections. A new query set which produces a constant number of relevant documents over the two collections is used to regulate the generality. The results are assessed from both the user and the system viewpoint; some strategies do appear equally effective on both collections.

1. Introduction

The ultimate goal of automatic information retrieval systems is to obtain a performance in "real life" situations equally as good as or better than in manual systems under operational conditions. Experiments done on automatic systems such as SMART are performed on controlled and limited collections. Therefore, in order to predict how the system will perform in a library situation, experiments on collections of different sizes are done and the results compared to see if there is a significant loss in performance as larger collections are used.

Generality is the proportion of relevant documents in a collection to total number of documents. In collections of varying sizes, generality is expected to differ, because the number of relevant documents does not increase proportionally to the number of nonrelevant documents. Therefore, results from test collections of different generality can be viewed as an indication of how the results from a test environment would be reflected in a real life situation.

This study is concerned with the relevance feedback aspect of information retrieval. Relevance feedback is one of the ways to utilize user opinion in improving search effectiveness [1]. A set of documents is given to the user who judges which documents are relevant to his request. This information is then used to modify his original query for another search through the collection. The rationale is that the original query might be badly worded, so that the incorporation of concepts from documents judged relevant might retrieve other related documents.

The method used in this study is to run several search strategies on collections of different generality and then to compare the retrieval performances. Several means are available to measure retrieval performance depending on the viewpoint taken. The recall-precision graph is used to represent the user viewpoint of how well the system is satisfying his needs. However, this is not adequate to measure system efficiency; consequently, fallout and adjusted precision have been developed. Fallout is the proportion of nonrelevant documents retrieved over total number of nonrelevant documents in the collection. When plotted against recall, this takes into account how much work the system has to do to retrieve equivalent numbers of relevant documents. When fallout is constant, precision can be adjusted to take generality into account so that the precision from collections of different generality can be compared on an equal basis [3].

2. Experimental Environment

The test collections should be similar in all respects except for generality. Ide [2] cites four factors which might account for the differences in results of the two collections she used — Cran 200 and ADI;

a) difference in subject matter
b) difference in collection scope
c) difference in variability within collection
d) difference in query construction and relevance judgment.

The CRN2NUL and CRN4NUL collections seem to eliminate these factors since they are subcollections of a homogeneous set — Cranfield 1400 — and are not mutually exclusive subcollections.

To vary the generality, the number of relevant items is held constant while the number of nonrelevant items varies. This can be done by creating a new query collection from the original CRN2NUL QUESTS and CRN4NUL QUESTS collections. The selected queries have the same relevance decisions in both the Cran 200 and Cran 400 collections. There are twenty-two such queries with a total of one-hundred and fifteen relevant documents. The formula for generality, averaged over all queries, is:

$$\text{Generality} = \frac{1000 \times \dfrac{\text{total relevant in collection}}{\text{number of queries}}}{\text{total number of documents in collection}} \qquad [4]$$

The generality for Cran 200 with respect to this new query set is 26.14 and for the Cran 400 is 12.30.

The query-update formula used for relevance feedback is:

$$Q_{i+1} = \pi Q_i + \omega Q_o + \alpha \sum_{1}^{\min(n_a, n_r')} r_i + \mu \sum_{1}^{\min(n_b, n_s')} s_i \qquad [2]$$

where	π, ω, α, μ	are multipliers
	Q_{i+1}	is updated query
	Q_i	is previous query
	Q_o	is original query
	n_r'	is number of relevant documents retrieved
	r_i	is relevant document retrieved
	n_s'	is number of nonrelevant documents retrieved
	s_i	is nonrelevant document retrieved
	n_a, n_b	specify the number of documents to be used.

Various strategies can be formulated using the above equation with the added parameters in the SEARCH routine of the SMART system, such as ALLOF, ATLEST and NOMOR. ALLOF is the number of documents to be retrieved. ATLEST is the minimum number of documents to be used in feedback and NOMOR is the maximum number of documents to be searched to provide documents for feedback. Only one iteration of feedback is used in this study because the most noticeable effect of feedback results from this iteration [5]. A frozen feedback iteration is used to eliminate the ranking effect for evaluation purposes.

Since the purpose of the experiment is to study the overall effect of feedback on these collections, a wide range of strategies are chosen:

Strategy 1 is positive feedback

Strategy is the "dec hi" strategy [2]

Strategy 4 is a modified "dec hi" strategy which uses a nonrelevant document for feedback only when no relevant documents are retrieved

Strategies 3 and 5 use varied multipliers.

The actual parameters are shown in Table 1.

This study attempts to determine whether feedback improves retrieval in one collection more than the other. That is, the initial full search results serve only as a base line and the improvement after using feedback is the result to be measured. Consequently, the following performance measures are stressed:

a) Precision improvement — $P_1 - P_0$
This indicates whether a particular strategy is better for one collection than the other from a user viewpoint.

		Strategies				
		1	2	3	4	5
Original Multiplier	ω	1	1	1	1	2
Positive Rank Cut	n_a (ALLOF)	10	5	5	5	5
Positive Multiplier	α	1	1	4	1	1
Negative Rank Cut	n_b (\$ALLOF)		1	5	1	5
Negative Multiplier	μ		-1	-1	-1	-1
Negative At Least	\$ATLST		1	0	1	0
Negative No More	\$NOMOR		10	5	5	5

Parameters for the Feedback Strategies

Table 1

P_0 is precision of initial search
P_1 is precision of feedback iteration
These are taken at fixed recall points.

b) Percentage of precision improvement — $\frac{P_1 - P_0}{P_0} \times 100\%$

This takes into account the fact that precision is better for a collection with a higher generality number [4] by taking the difference with respect to the original precision.

c) Fallout improvement — $F_0 - F_1$

A performance improvement implies that fallout for the feedback iteration is less than fallout for the initial search. This equation is equivalent to $(-1) \times (F_1 - F_0)$ and multiplying by -1 serves to transform the difference onto the positive scale.

F_0 is fallout of initial search
F_1 is fallout of feedback iteration
These are taken at fixed recall points.

d) Percentage of fallout improvement — $\frac{F_0 - F_1}{F_0} \times 100\%$

This takes into account the fact that the fallout is not the same for the initial searches on both collections. Therefore, the difference is computed as a percentage of the original.

e) Adjusted precision — $P_A = \frac{R_1 \times G_2}{(R_1 \times G_2) + F_1(1000 - G_2)}$ [4]

Precision of the Cran 200 is adjusted to that of Cran 400 and not vice versa, because the emphasis of this study is on performance of larger collections.

R_1 is fixed recall points
F_1 is fallout of Cran 200 at R_1 recall
G_2 is generality of Cran 400

In this manner, the results from two collections of different generality can be compared on an equal basis. This comparison is from a system viewpoint.

f) Adjusted precision improvement — $P_{A_1} - P_{A_0}$

Similar to a).

g) Percentage of adjusted precision improvement —

$$\frac{P_{A_1} - P_{A_0}}{P_{A_0}} \times 100\%$$

Similar to b).

3. Experimental Results

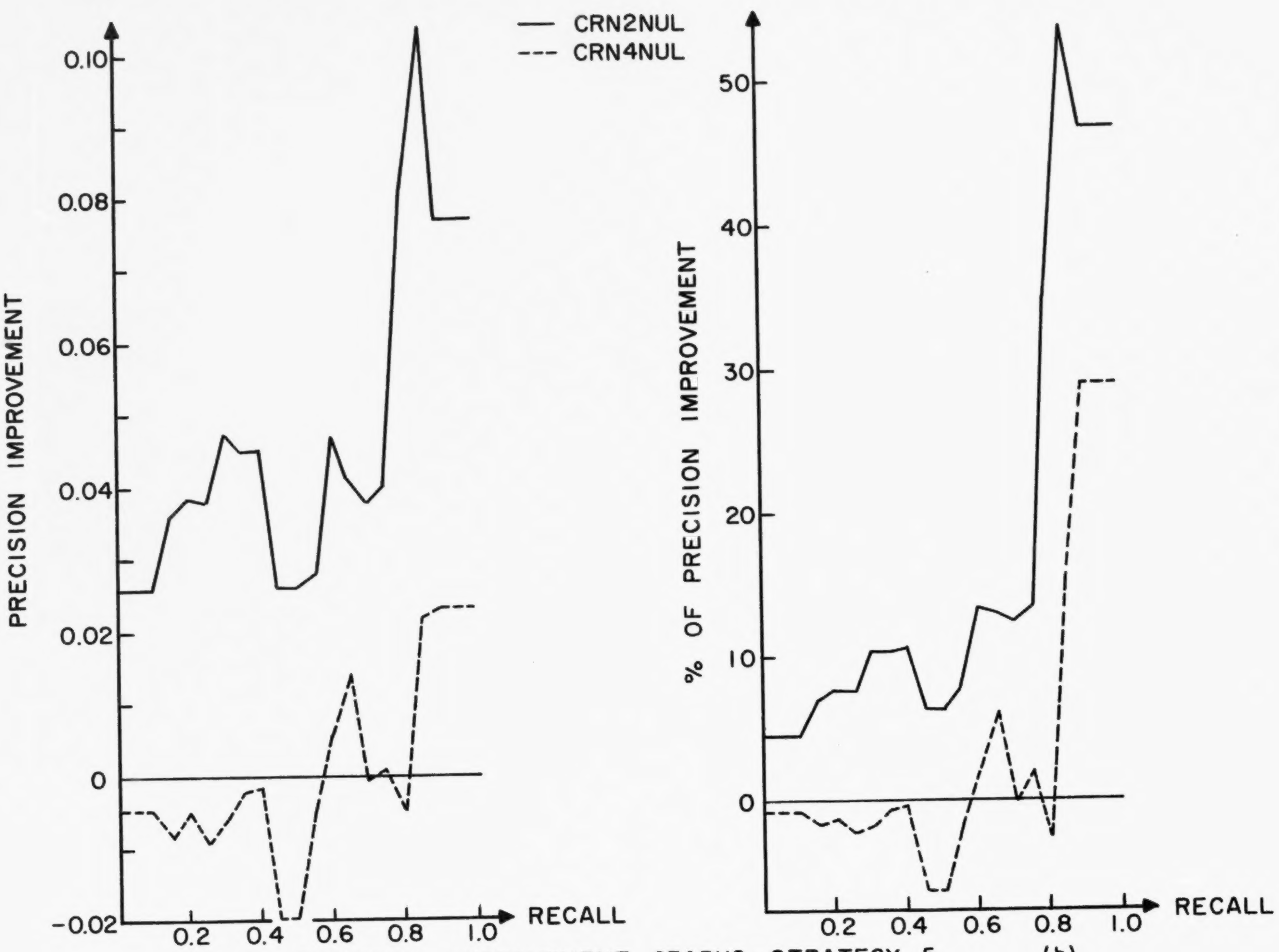
The results seem to fall into two categories with strategies 2, 3 and 5 in one group (group A) and strategies 1 and 4 in the other group (group B). The former group consistently shows a good performance for Cran 200, but there is little improvement for Cran 400, whereas the latter group shows an equivalent improvement. The average improvements for one strategy from each group are shown in Table 2.

In group A from both a system and a user viewpoint, the Cran 200 performs better as can be seen in all the improvement graphs. In fact, for strategy 5, Cran 400 performs worse using feedback than for the full search as shown by the negative values in the precision improvement curve (Fig. 1a) and the percentage fallout improvement curve (Fig. 2b). This result seems to indicate that this class of feedback strategies will not perform well in a library situation. For strategies 3 and 5, the result is probably due to the large number of nonrelevant documents

	Strategy 1		Strategy 3	
	Cran 200	Cran 400	Cran 200	Cran 400
Precision improvement	.0293	.0307	.0526	.0218
% of Precision improvement	12.50	16.99	19.38	12.73
Fallout improvement	.0104	.0770	.0130	.0066
% of Fallout improvement	13.38	15.94	21.73	12.16
Adj Precision improvement	.0194		.0379	
% Adj Precision improvement	24.21		23.79	

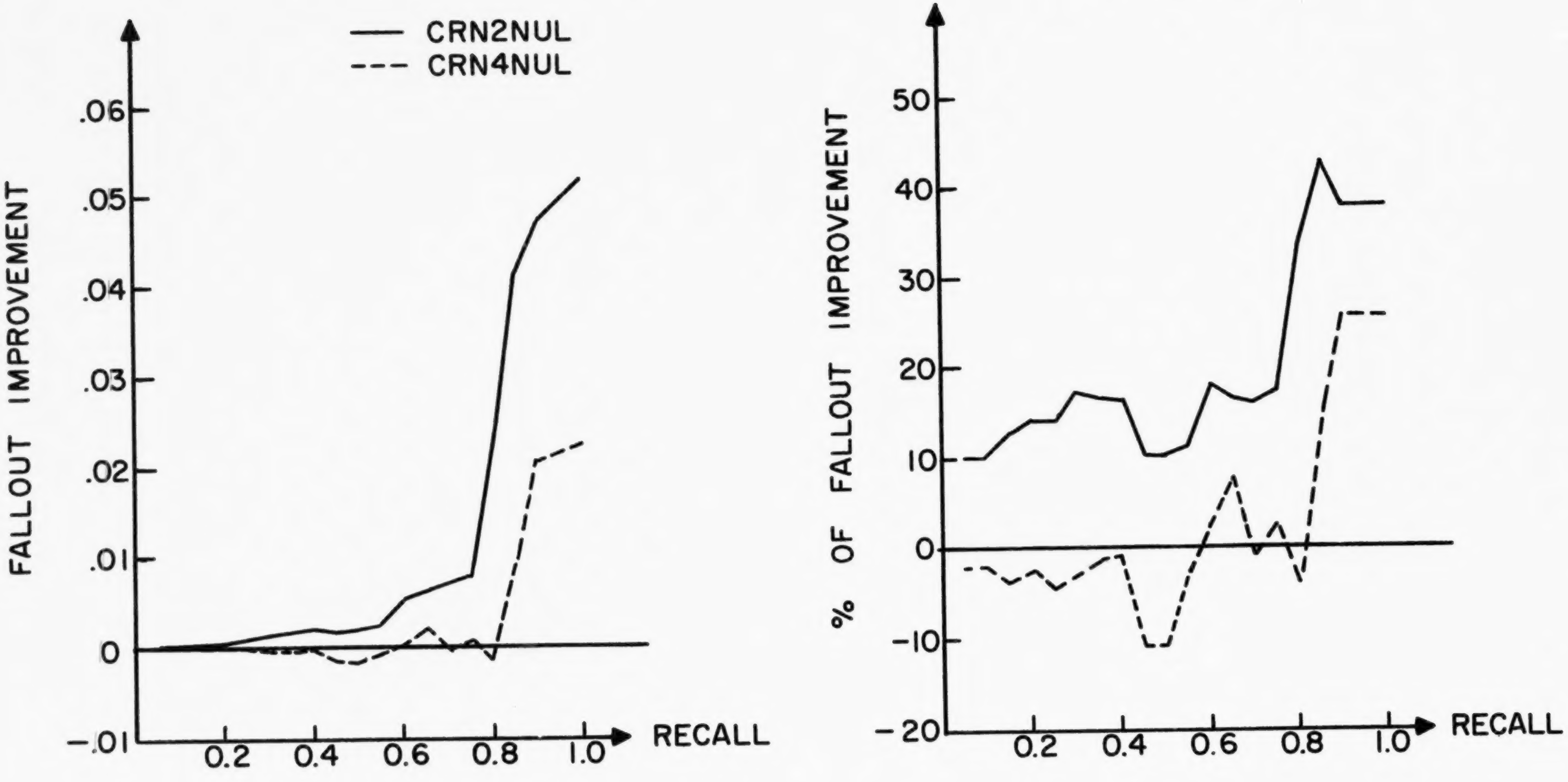
Average Improvement Results for Strategies 1 & 3

Table 2



(a) PRECISION IMPROVEMENT GRAPHS-STRATEGY 5 (b)

Figure 1



(a) (b)

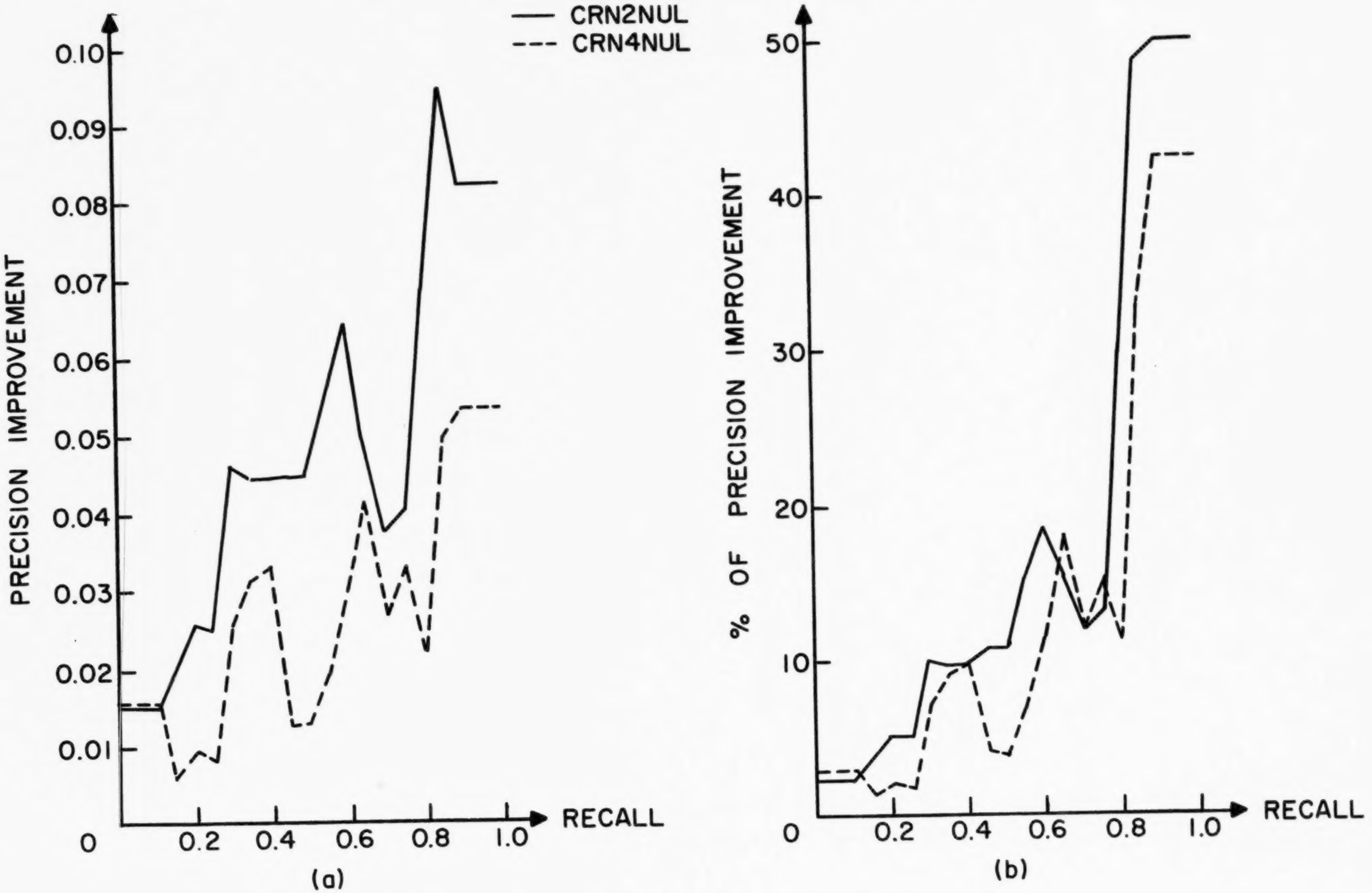
FALLOUT IMPROVEMENT GRAPHS
STRATEGY 5

Figure 2

used for feedback which tend to eliminate the query. There is a median of four nonrelevant documents used for feedback on the Cran 400 and of three documents on the Cran 200. In strategy 5 on the Cran 400, out of the twenty-two queries, seven queries have two or fewer concepts left after feedback whereas on the Cran 200 there are only four such queries. The larger multiplier for the original query in strategy 3 partially offsets this effect of erasing the query.

As for strategy 2 which always uses one nonrelevant document for feedback, the Cran 200 precision improves while the Cran 400 precision does not (Fig. 3a). This is due to the fact that on the Cran 200 there would be more relevant documents retrieved (median of 2); therefore, one nonrelevant document does not erase the query. On the Cran 400, however, fewer relevant documents would be retrieved (median of 1); therefore, one nonrelevant document might remove more concepts than are added by the relevant documents in the feedback.

Looking at the precision improvement graphs for group B, Cran 200 and Cran 400 curves using strategy 1 (Fig. 4a) are interspersed whereas for strategy 4, the Cran 200 curve is usually higher (Fig. 5a). But looking at the percentage precision graphs, for strategy 1 (Fig. 4b), the Cran 400 is better at all recall points. This is not unexpected, since the original precision of the Cran 400 is lower than that of the Cran 200. Therefore even with a similar increase in precision, from a system viewpoint, the feedback is more helpful in improving retrieval for the Cran 400 since this larger collection

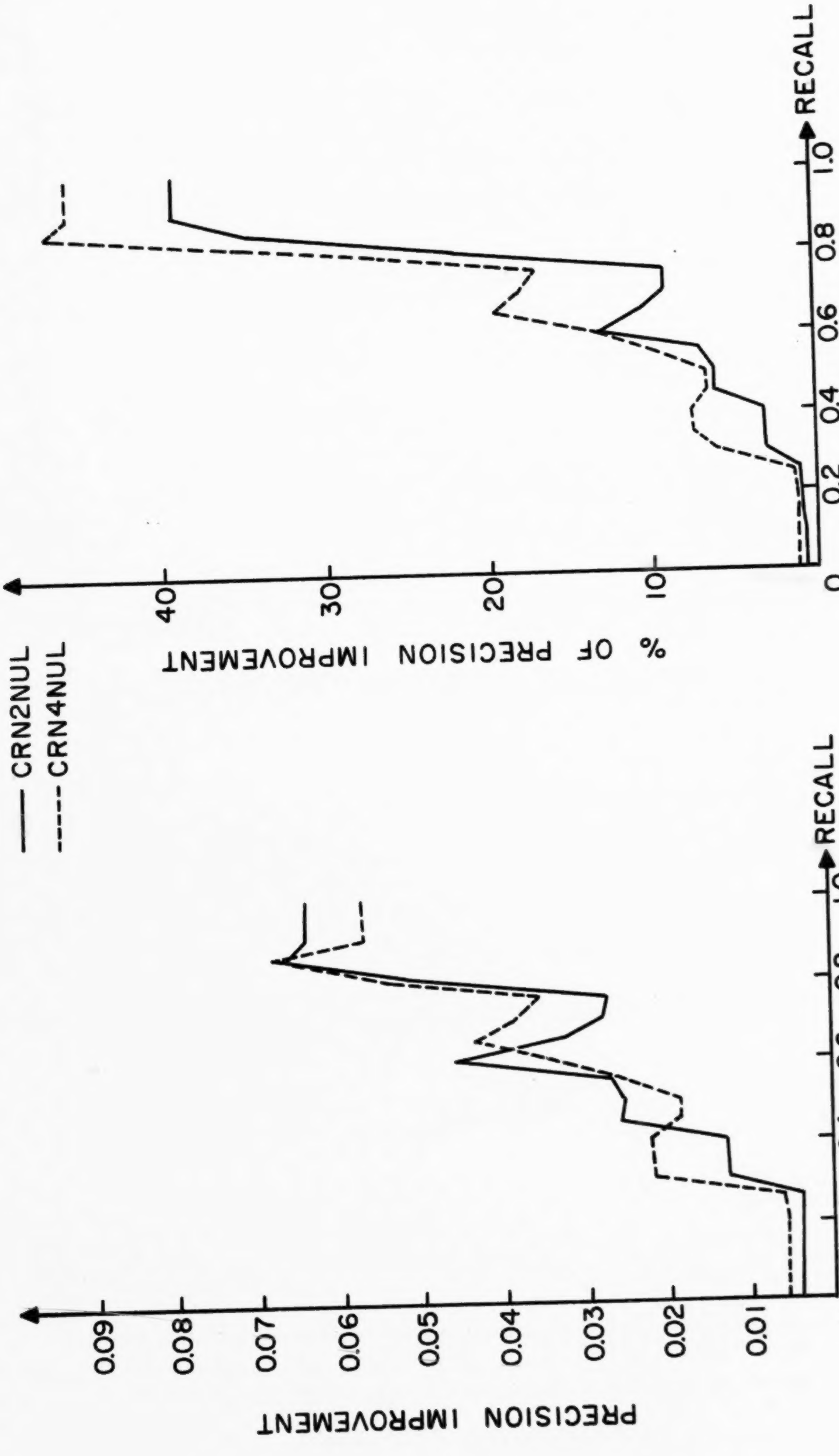


(a)

(b)

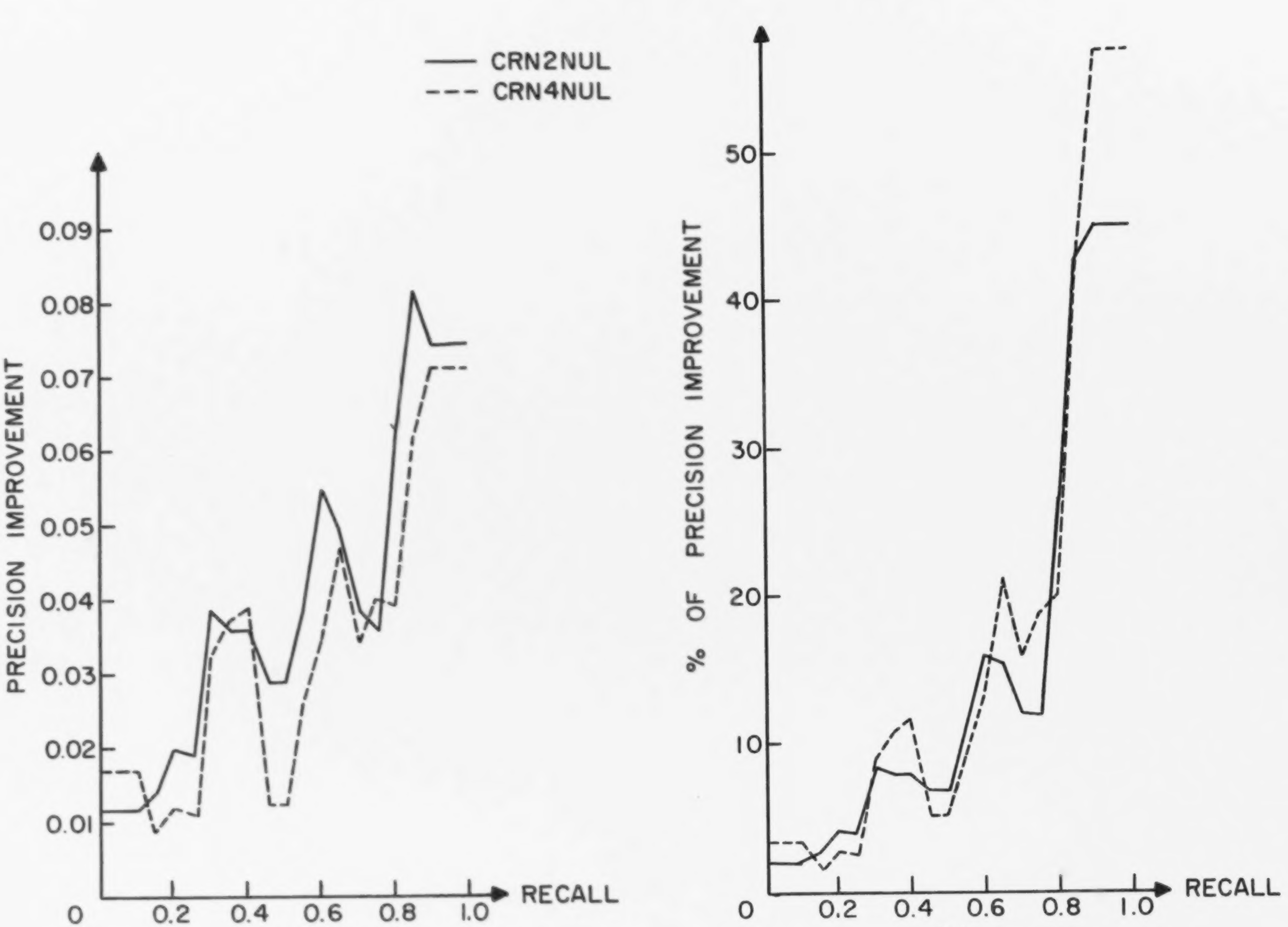
PRECISION IMPROVEMENT GRAPHS - STRATEGY 2

Figure 3



PRECISION IMPROVEMENT GRAPHS – STRATEGY I

Figure 4



(a)

(b)

PRECISION IMPROVEMENT GRAPHS – STRATEGY 4

Figure 5

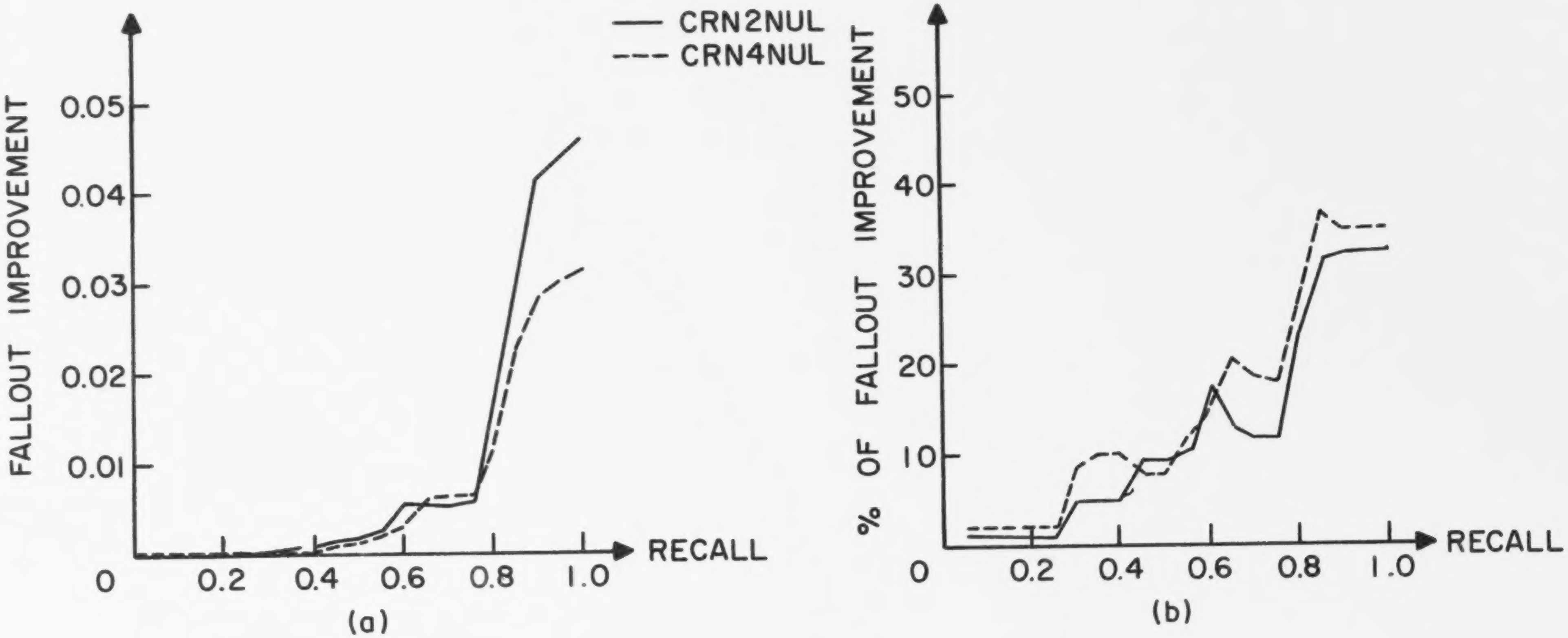
is not as favorable to retrieval as a smaller collection in the first place (lower original precision). For strategy 4 (Fig. 5b), the two curves are interspersed instead of the Cran 400 being lower because once the original precision is taken into account, the percentage increase becomes similar.

Theoretically, the fallout curves (see Appendix) for the two collections should be the same. However, there is probably a subset in the Cran 400 collection of nonrelevant documents which have a very low probability of being retrieved [4]. This explains why fallout for the Cran 400 seems better, a fact to be remembered when comparing fallout values.

For strategy 1, in the fallout improvement graph (Fig. 6a), Cran 200 is for the most part better. On the corresponding percentage fallout improvement graph (Fig. 6b) the Cran 400 is slightly better. For stragety 4, on the other hand, the difference in fallout improvement is more pronounced and the percentage fallout improvement is more similar (Fig. 7a, 7b).

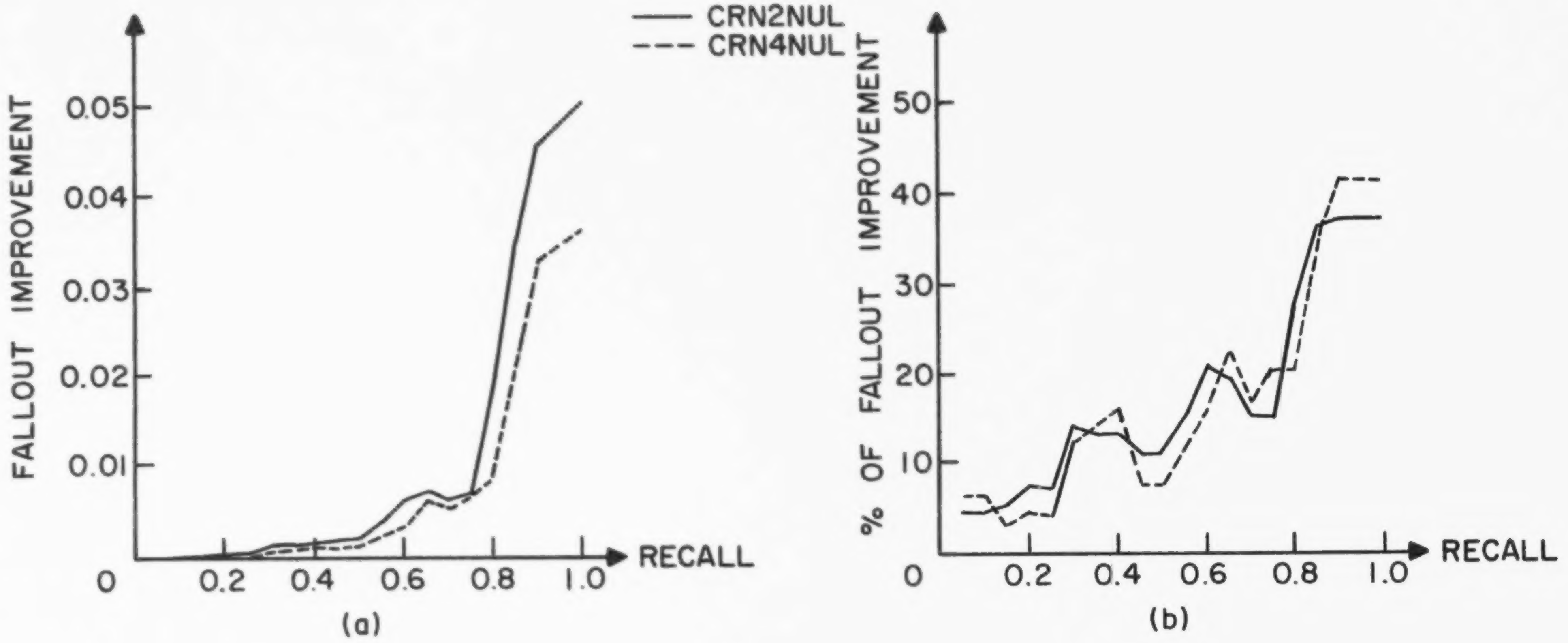
These fallout results are quite logical. Since on the feedback run, the number of relevant documents retrieved on the Cran 200 tends to be larger than for the Cran 400 (usually one more relevant document for Cran 200), the number of nonrelevant documents would be smaller. Therefore, the fallout improvement for the Cran 200 is larger. However, when the original fallout values are considered, the two collections become similar.

Once precision for the Cran 200 is adjusted to that of the Cran 400, the recall-precision curve for the Cran 400 is



FALLOUT IMPROVEMENT GRAPHS
STRATEGY I

Figure 6



FALLOUT IMPROVEMENT GRAPHS
STRATEGY 4

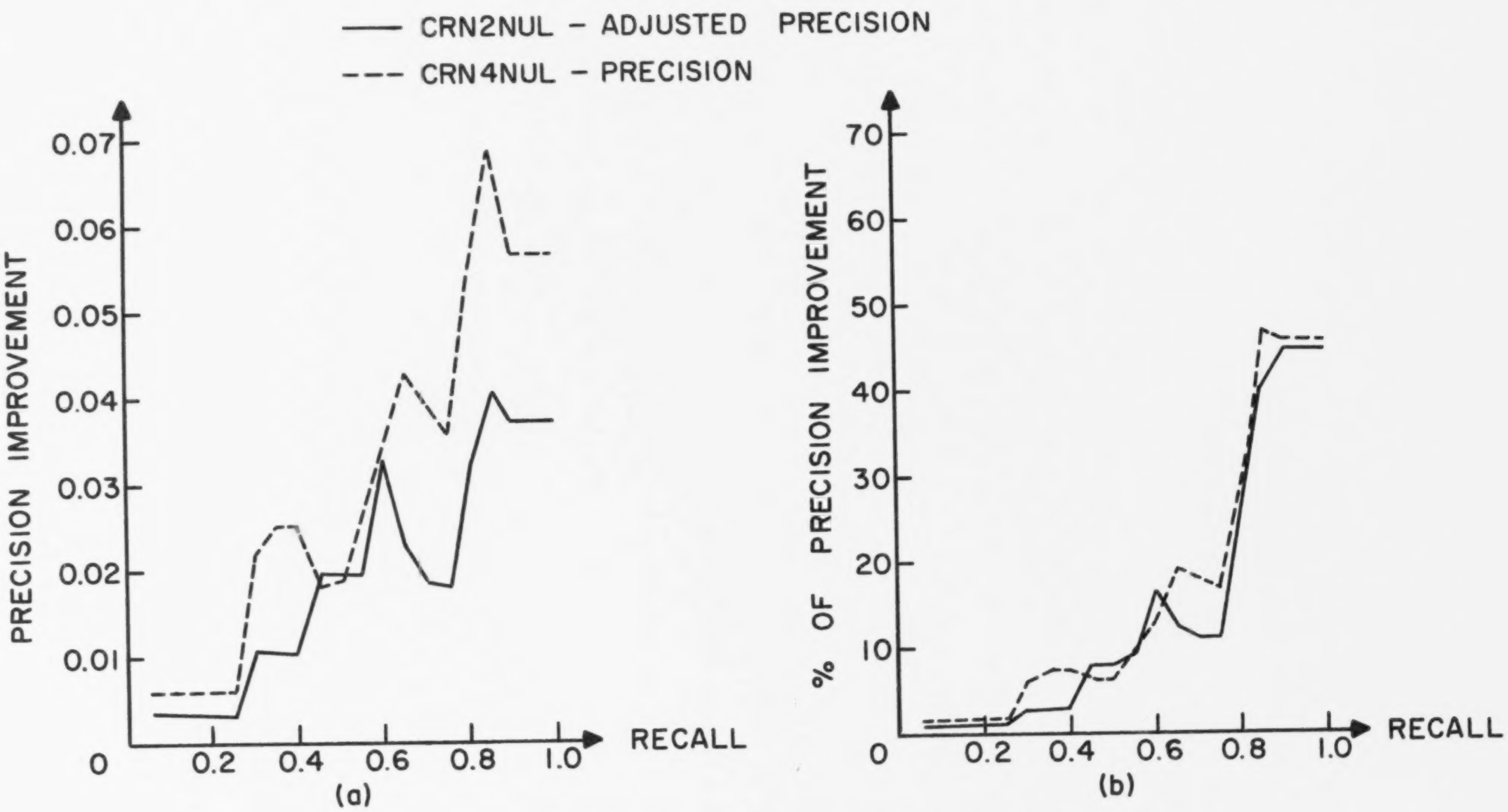
Figure 7

is lower than that for the Cran 400 (see Appendix). Therefore, according to these graphs, from a system viewpoint, Cran 400 definitely shows better performance. From the adjusted precision improvement graphs (Fig. 8a, 9a), the improvement of Cran 400 is at least equal if not more than that of Cran 200. This result is also supported by the percentage adjusted precision improvement graphs (Fig. 8b, 9b). From both a user and a system viewpoint, it would appear that use of these feedback strategies is at least as effective for a larger collection (lower generality number).

An interesting comparison can be made between strategies 2 and 4 since they are similar in that both use negative feedback of one nonrelevant document. However, the fact that strategy 4 uses negative feedback only when no positive feedback can be performed, as opposed to strategy 2 which uses it for all queries, causes strategy 4 to be effective and strategy 2 to fail on the Cran 400. For strategy 4, the few relevant documents used in feedback are not offset by any negative feedback as they would be for strategy 3 (see discussion of strategy 2 above).

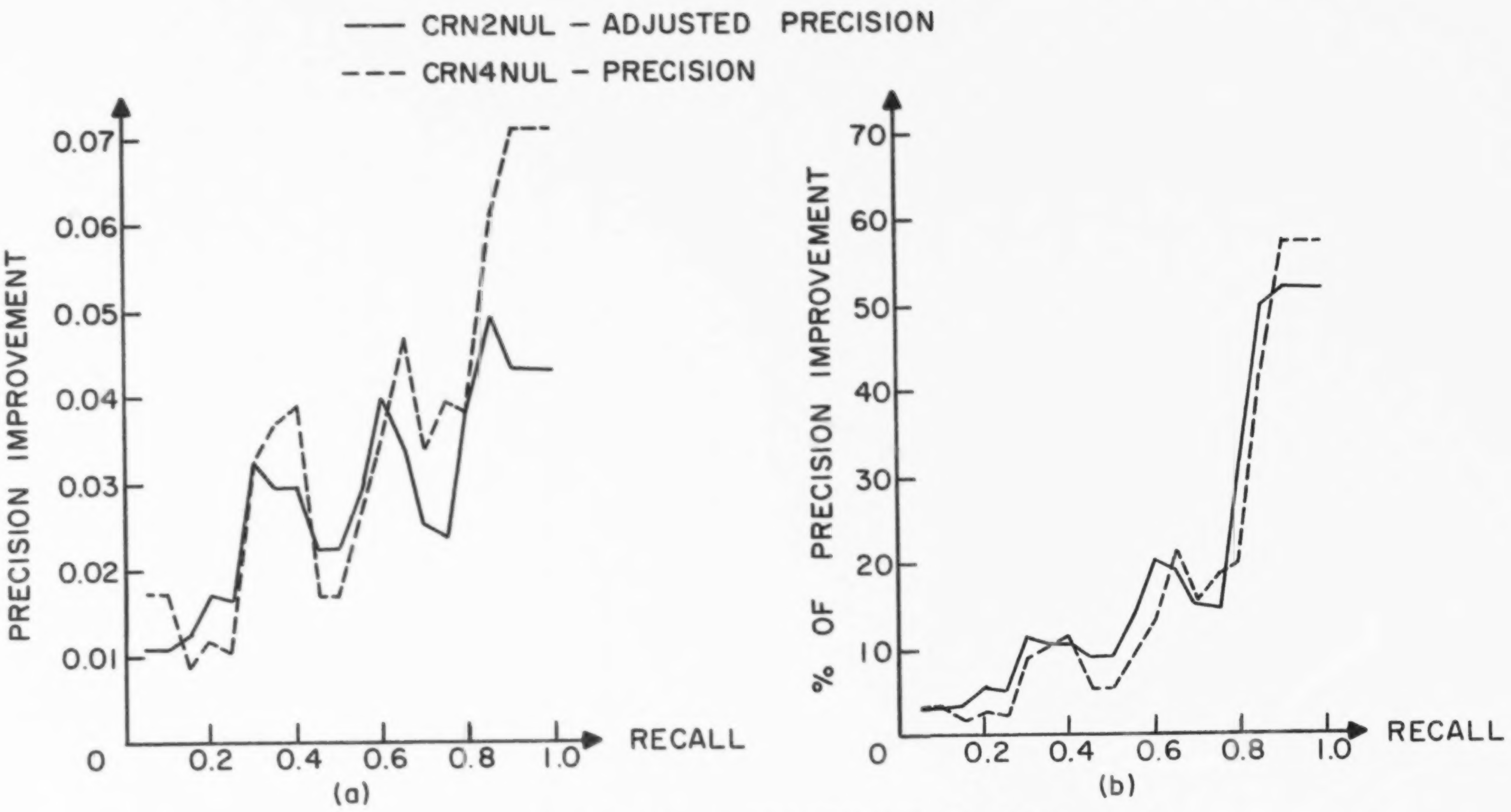
4. Conclusion

Results of this study are encouraging in that they seem to indicate that some feedback strategies can indeed be used in a realistic environment. Those commonly used strategies such as pure positive feedback and the strategy which uses the top ranking nonrelevant document only when no relevant documents are



PRECISION IMPROVEMENT GRAPHS
WITH PRECISION OF CRAN 200 ADJUSTED
STRATEGY I

Figure 8



PRECISION IMPROVEMENT GRAPHS
WITH PRECISION OF CRAN 200 ADJUSTED
STRATEGY 4

Figure 9

retrieved, are equally effective on the Cran 200 and the Cran 400.

It is generally believed that feedback on a collection of lower generality will not be as effective and that feedback on a collection as large as a library is not promising. However, the results of this study to seem to point out that relevance feedback would be operative on a library collection, contrary to common belief. Of course this is highly dependent on which feedback method is used, since some strategies (such as those using a large number of nonrelevant documents) perform poorly on collections of lower generality. Furthermore, as the fallout curves indicate, the Cran 400 collection might have a disjoint subset of documents never retrieved. Thus generality should be recomputed by removing such documents. In addition, the test collections used here are limited in that they pertain to only one subject area.

A suggestion for future experiments is that queries should be examined individually to isolate irregular behavior. Also a larger query collection and document collection on more than one subject area would be advisable to substantiate the results. Based on the findings of this study, variations of the two feedback strategies in group B — e.g. requiring a constant number of relevant documents to be fed back or using different rank cut values — should be explored.

References

[1] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill Book Company, New York, 1968, pp. 266-267.

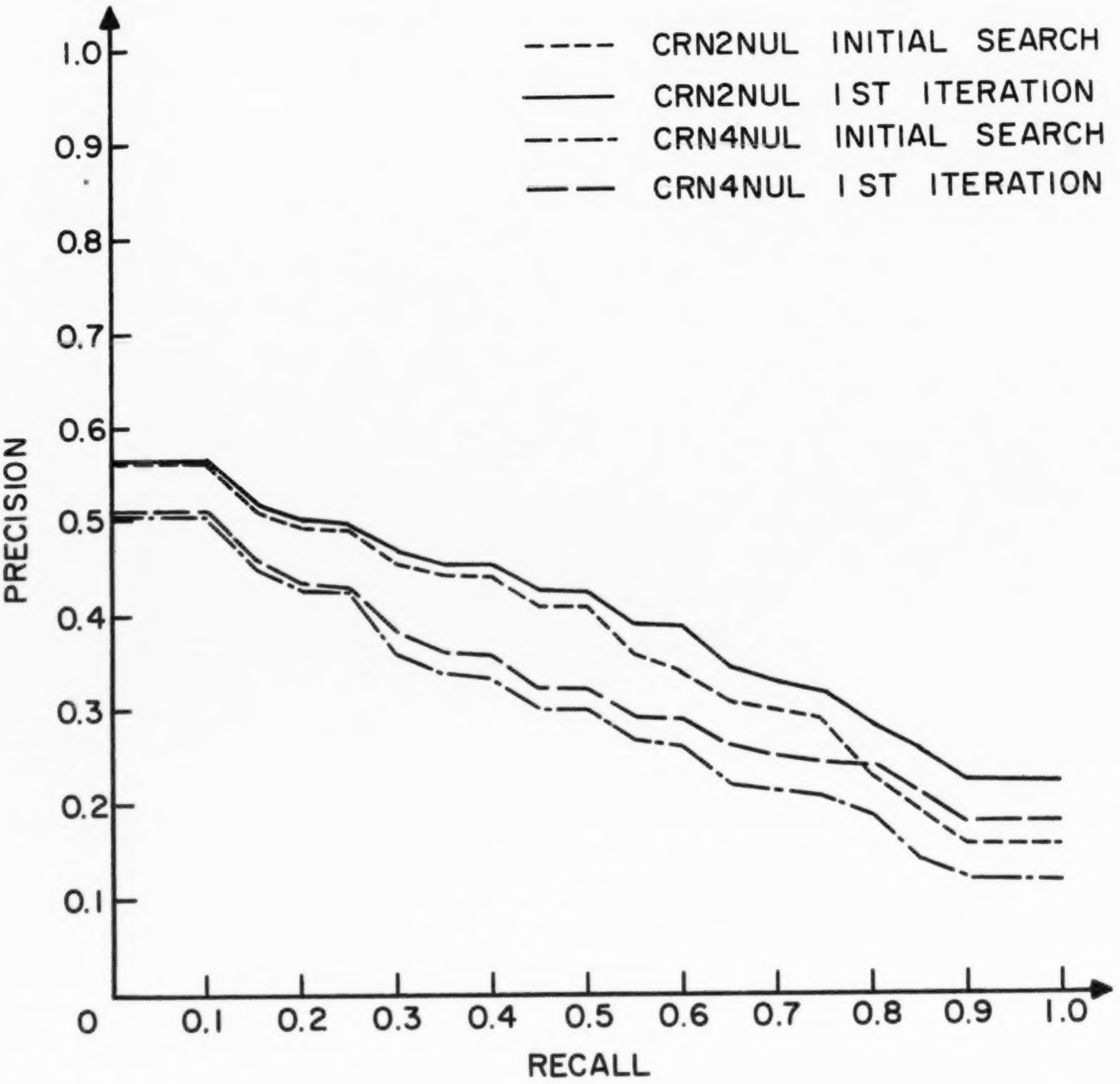
[2] E. Ide, User Interaction with an Automated Information Retrieval System, Report No. ISR-12 to the National Science Foundation, Section VIII, Department of Computer Science, Cornell University, June 1967.

[3] E.M. Keen, Evaluation Parameters, Report No. ISR-13 to the National Science Foundation, Section II, Department of Computer Science, Cornell University, January 1968.

[4] C. Cleverdon and M. Keen, Factors Determining the Performance of Indexing Systems, Vol. 2 Test Results of the ASLIB Cranfield Research Project, C. Cleverdon, Wharley End, Bedford, 1966.

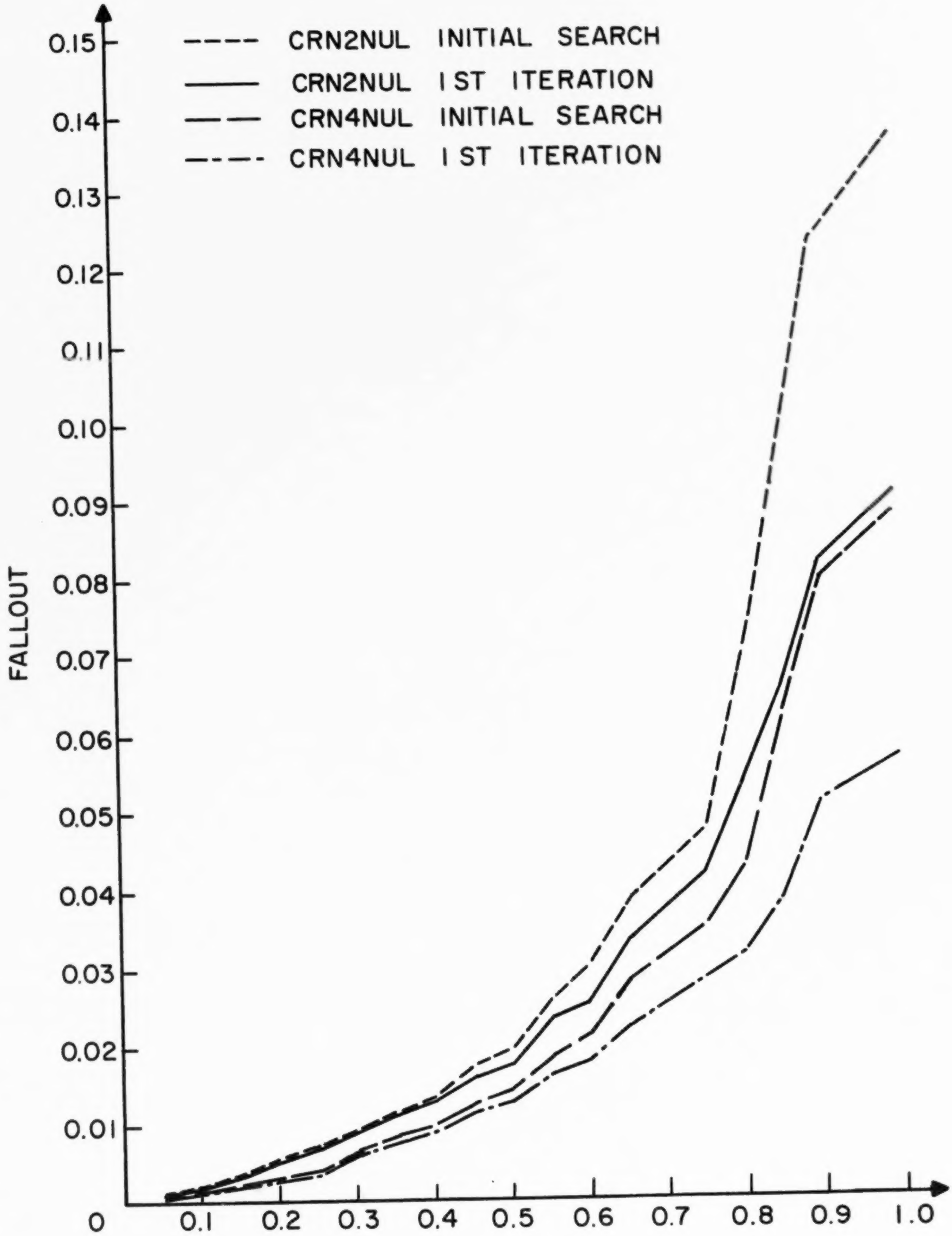
[5] G. Salton, Search Strategy and the Optimization of Retrieval Effectiveness, Report No. ISR-12 to the National Science Foundation, Section V, Department of Computer Science, Cornell University, June 1967.

Appendix



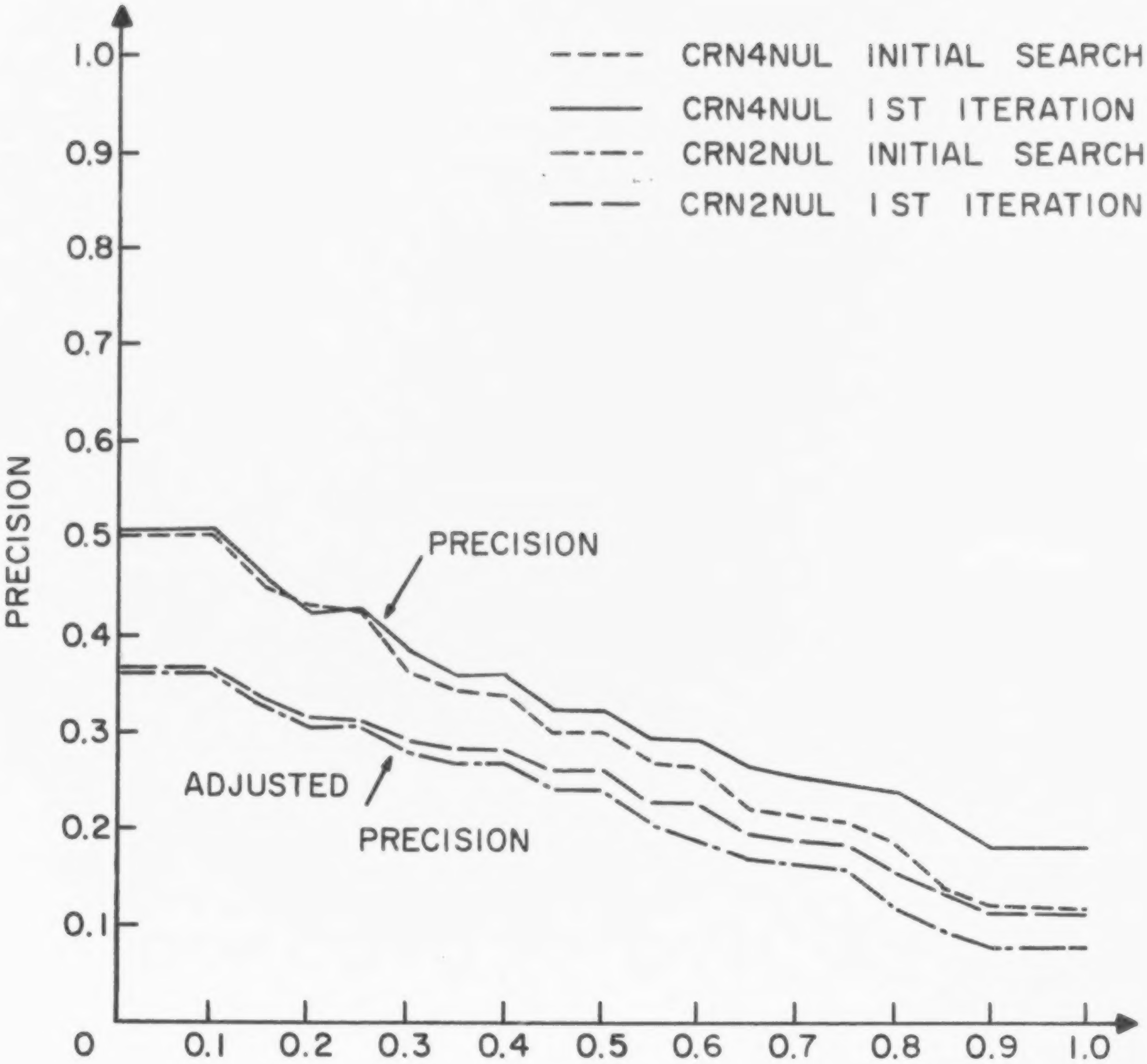
STRATEGY I

RECALL-PRECISION GRAPH



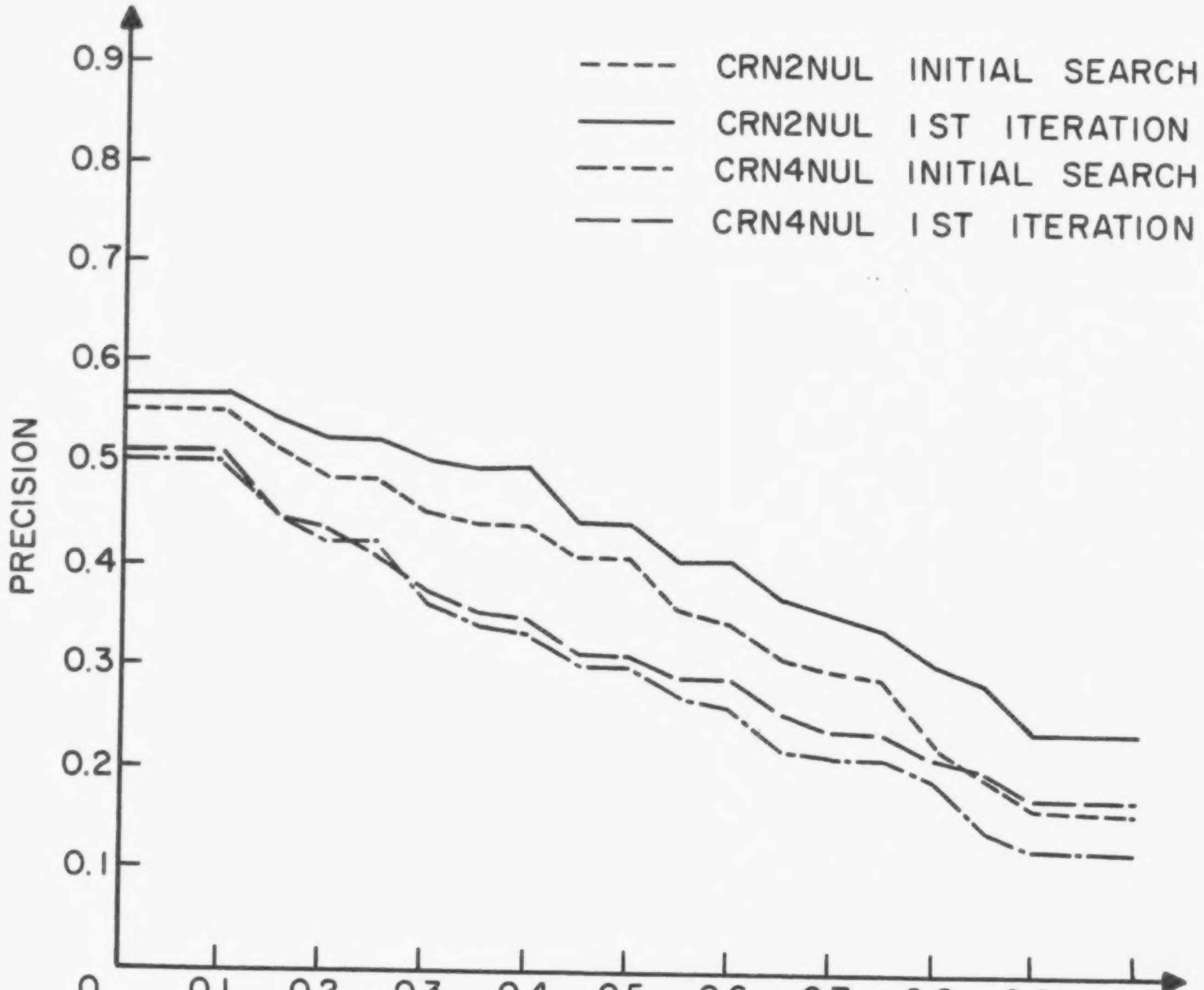
STRATEGY I

RECALL-FALLOUT GRAPH



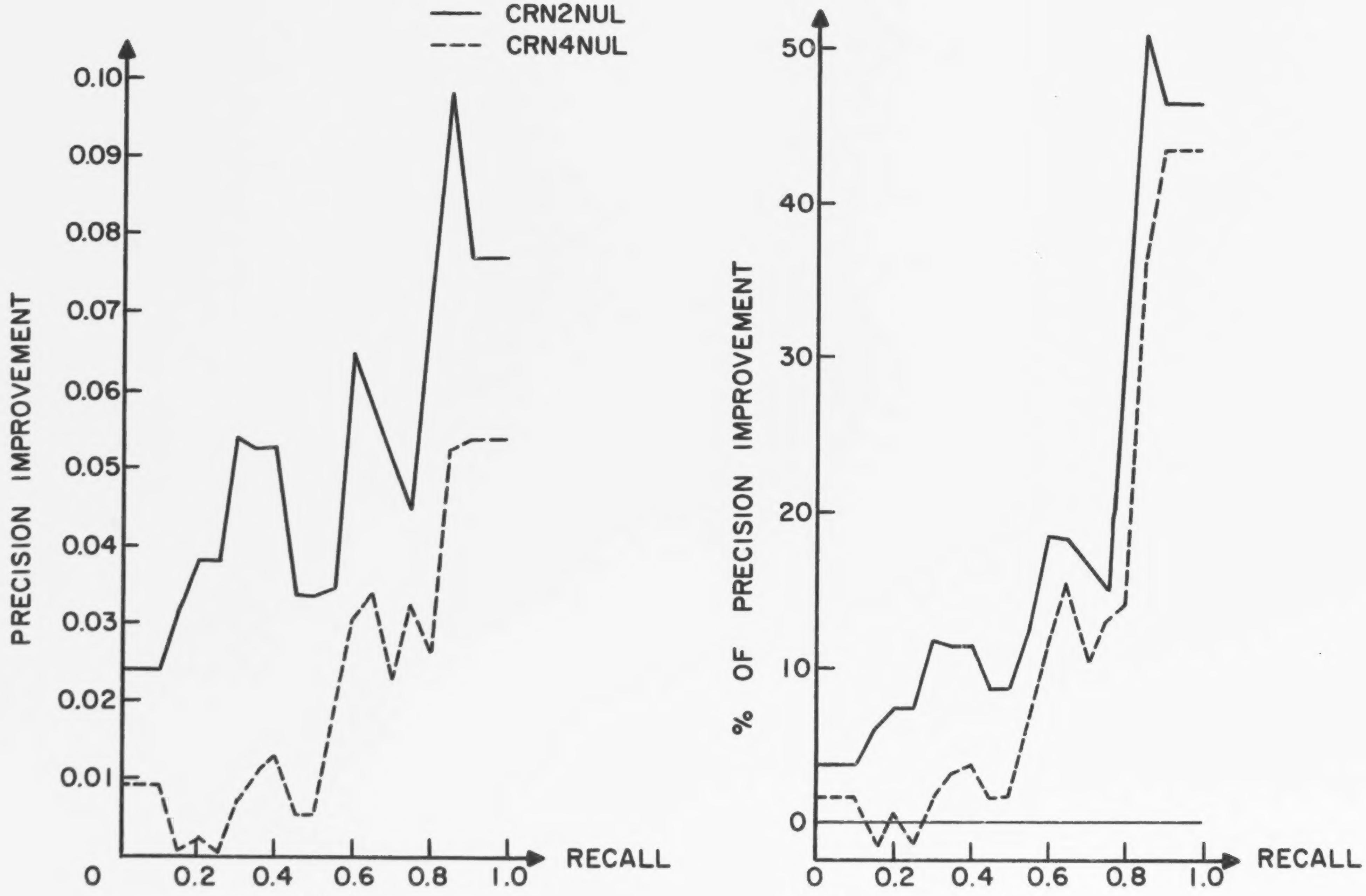
STRATEGY 1

RECALL-PRECISION GRAPH WITH PRECISION OF CRAN 200 ADJUSTED



STRATEGY 3

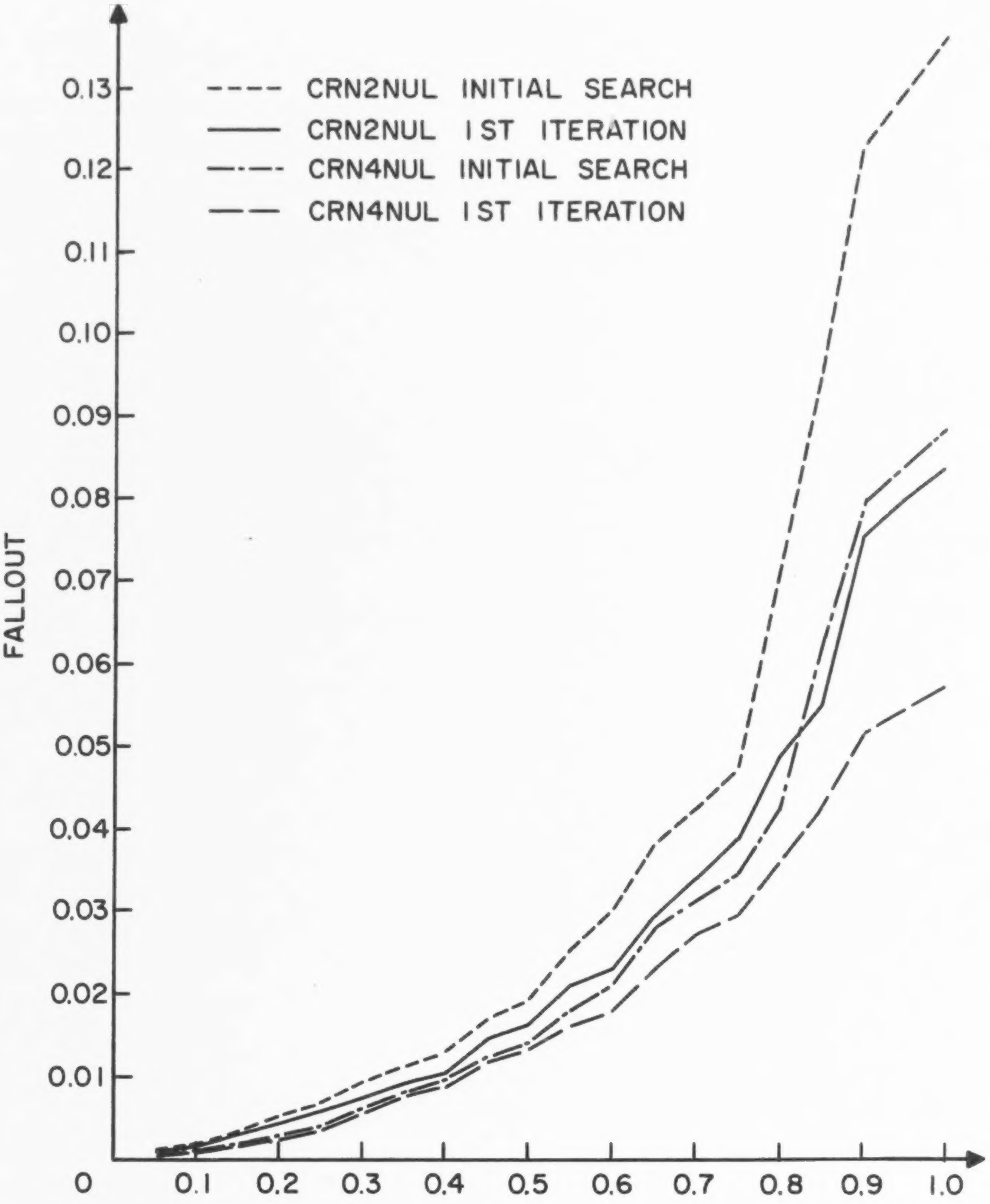
RECALL-PRECISION GRAPH



(a)

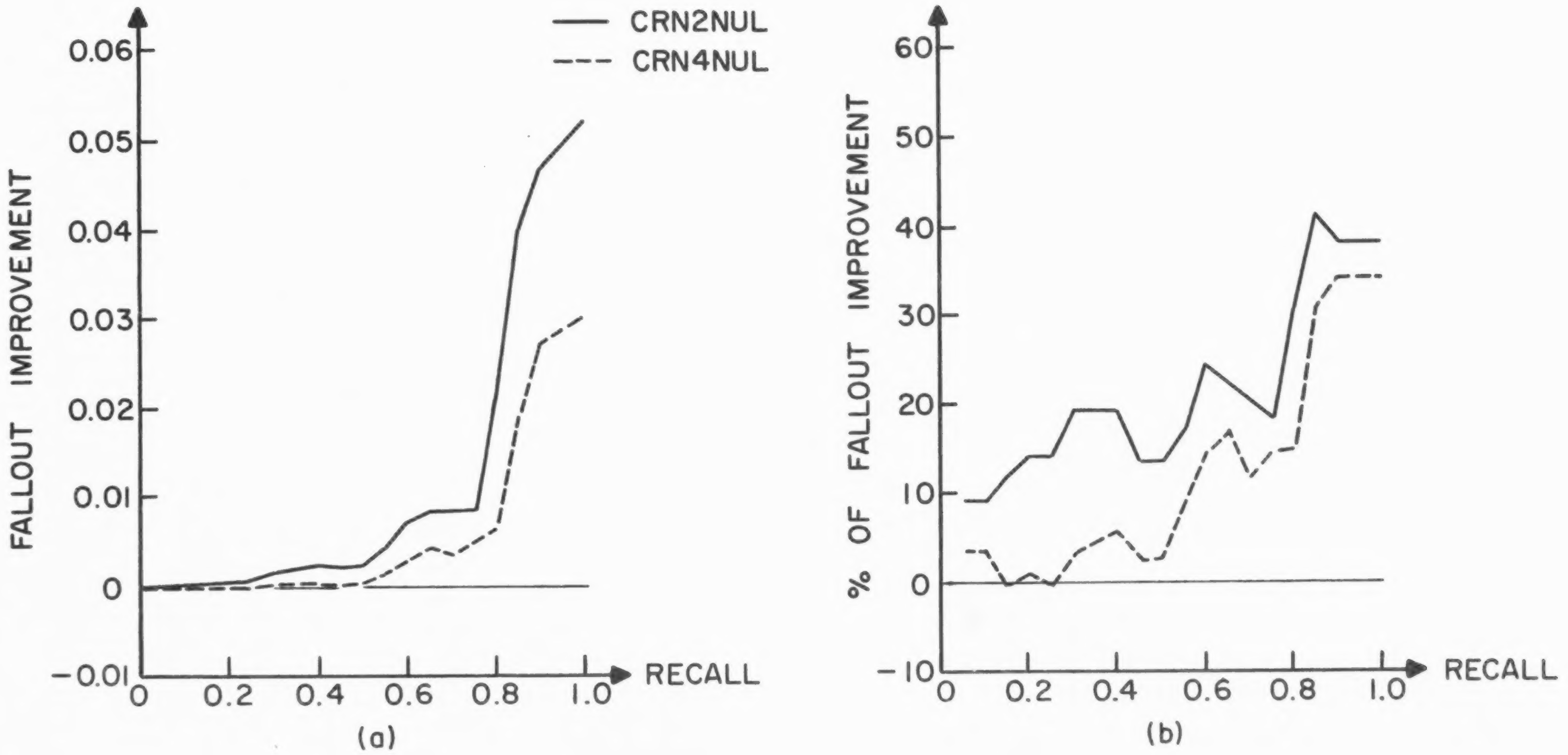
(b)

PRECISION IMPROVEMENT GRAPHS
STRATEGY 3



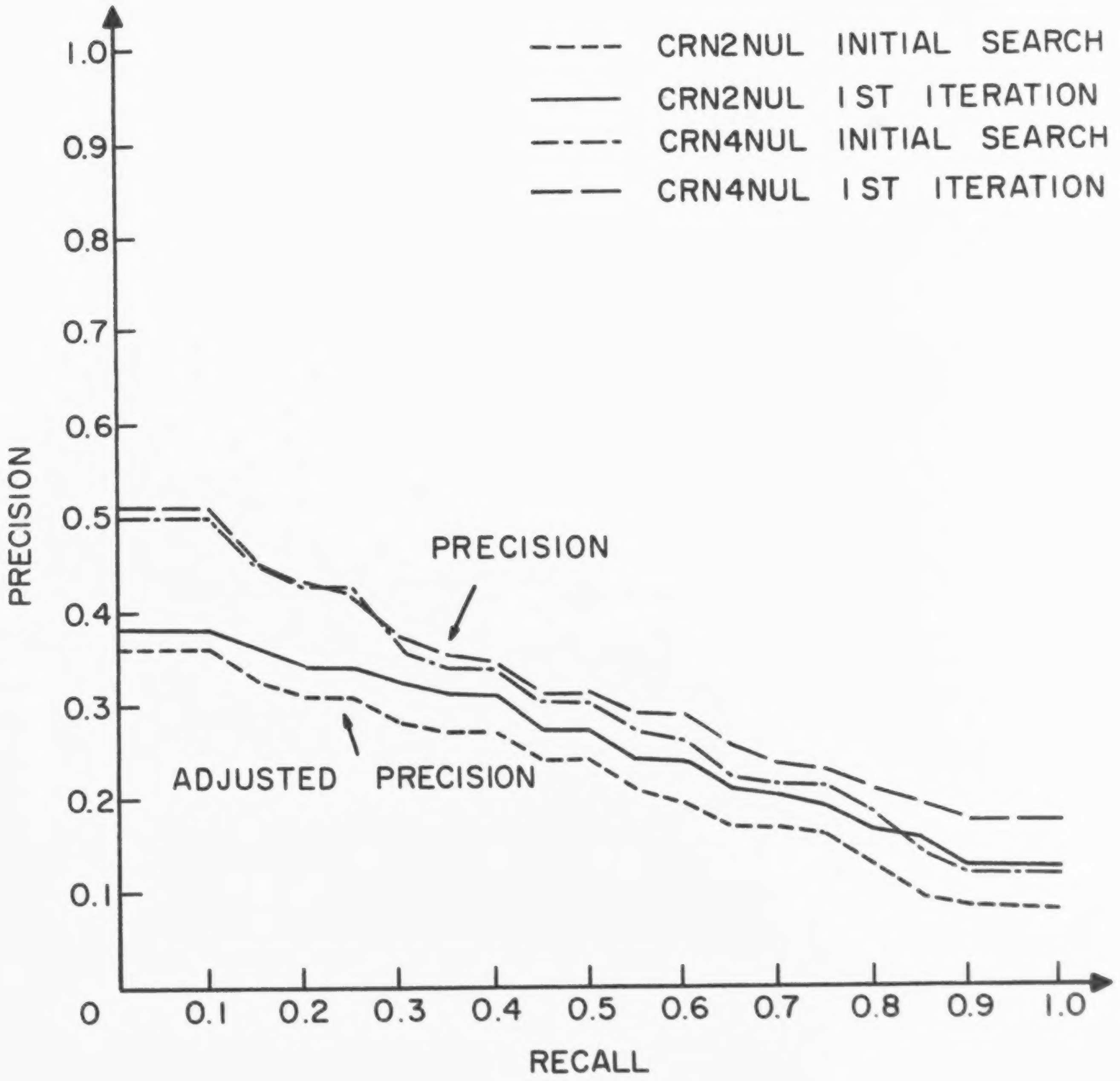
STRATEGY 3

RECALL — FALLOUT GRAPH



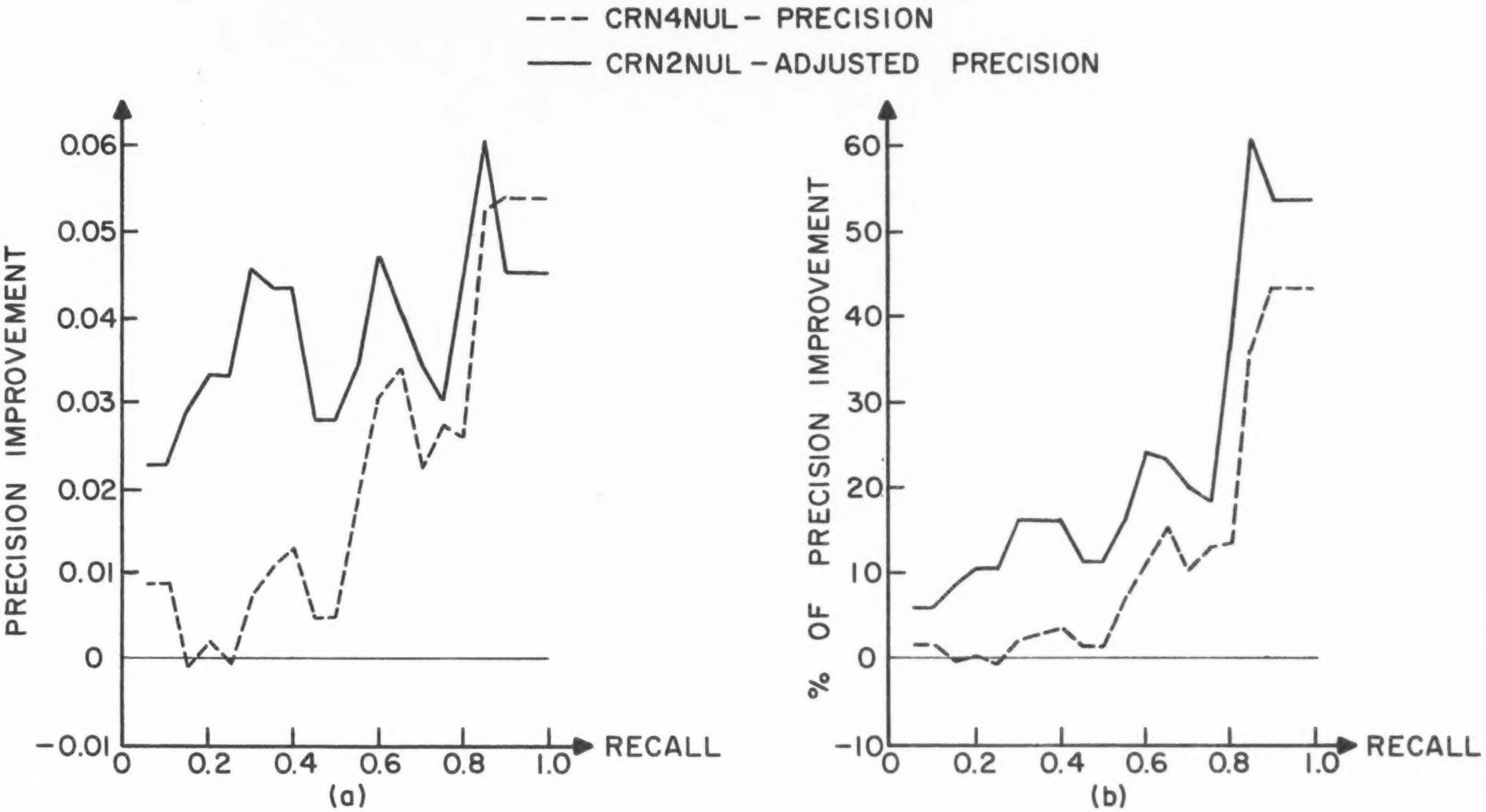
FALLOUT IMPROVEMENT GRAPHS

STRATEGY 3



STRATEGY 3

RECALL-PRECISION GRAPH WITH
PRECISION OF CRAN 200 ADJUSTED



(a)

(b)

PRECISION IMPROVEMENT GRAPH

WITH PRECISION OF CRAN 200 ADJUSTED

STRATEGY 3

X. Selective Negative Feedback Methods

M. Kerchner

Abstract

A great deal of work has already been done in automatic information retrieval in an effort to improve performance and to satisfy user needs. In particular various techniques have been described which modify the initial query submitted by the user, including the use of nonrelevant and relevant retrieved documents. The present study deals with experiments performed with several new methods of using nonrelevant retrieved documents to modify queries which retrieve no relevant in the first N documents retrieved. The results of the experiments are evaluated and suggestions are made for possible further investigations.

1. Introduction

Relevance feedback is a technique for improving the performance of an information retrieval system to better satisfy the needs of its users. [1] A search of the document collection is made with an initial query and a set of retrieved documents, ranked in order of correlation with the query, is presented to the user. After examining the set of retrieved documents, the user indicates whether each is relevant or not relevant to his query. [3] The relevance judgments are used by the system to modify the original search query in such a way that the modified query will retrieve additional relevant documents.

Experiments have been made with several methods of positive relevance feedback in which highly ranked relevant documents are used to modify the query. [2,4] In the case where no relevant documents are retrieved in the first N documents considered, negative relevance feedback — the use of nonrelevant documents for query modification — has been the basis for experimentation. [1,2,5] However, some problems arise with the use of nonrelevant documents for query modification. Riddle et. al. [4] and Ide [5] confirm that in some cases the use of nonrelevant documents perturbs the query vector so grossly that no additional relevant documents are retrieved in subsequent searches with the modified query. [6]

In the present study, the SMART document retrieval system is used as the basis for experiments on methods which propose to deal with the above and related problems.

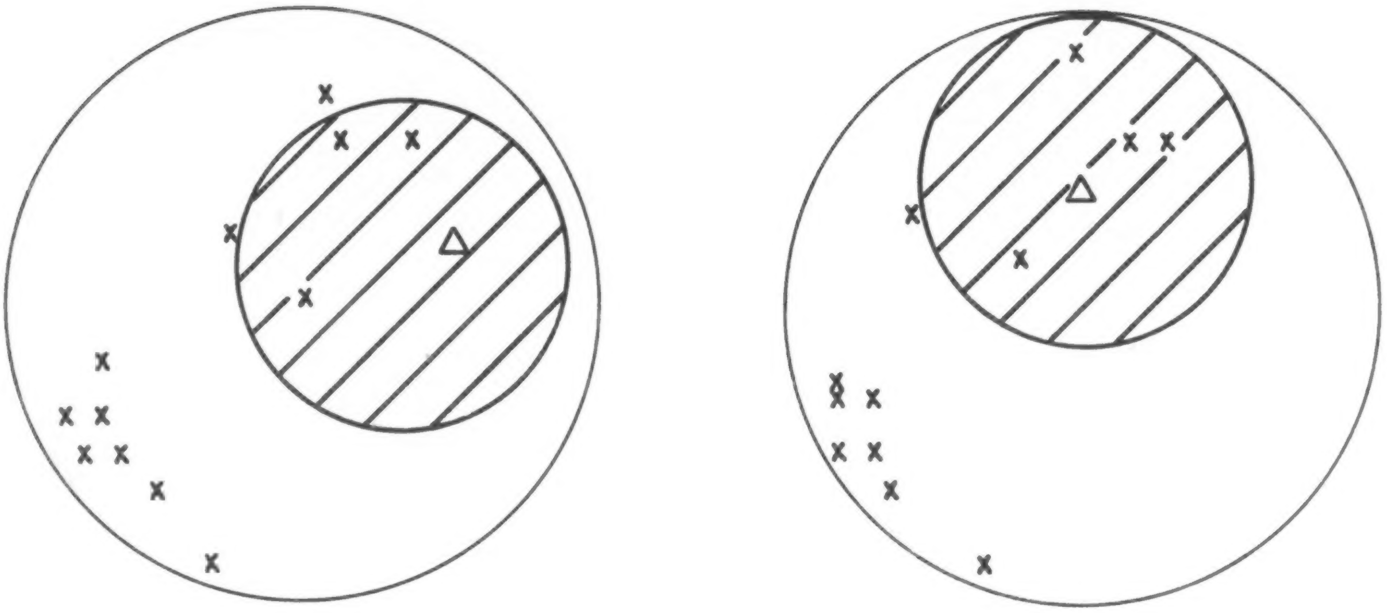
2. Methodology

It has been shown by previous work that methods using positive relevance feedback are reasonably successful for queries retrieving at least one relevant document in the first N retrieved. Therefore, the experiments in this study are only concerned with those queries which retrieve no relevant in the first N (N=5) documents retrieved.

To deal with the problem of overdistortion of the query which occurs with standard negative feedback schemes in which highly ranked nonrelevant documents are subtracted from the query, Johnson and Krablin [6] propose that more selective methods be used in order to "insure the integrity of the original relevant concepts in the query" and to move the query out of an area of nonrelevant concepts in the document space by using a series of selected terms for negative feedback. The approach suggested by Johnson and Krablin is to select those terms which appear in several of the highly correlated nonrelevant documents, but not in the original query and to add these terms, with negative weights, to the query.

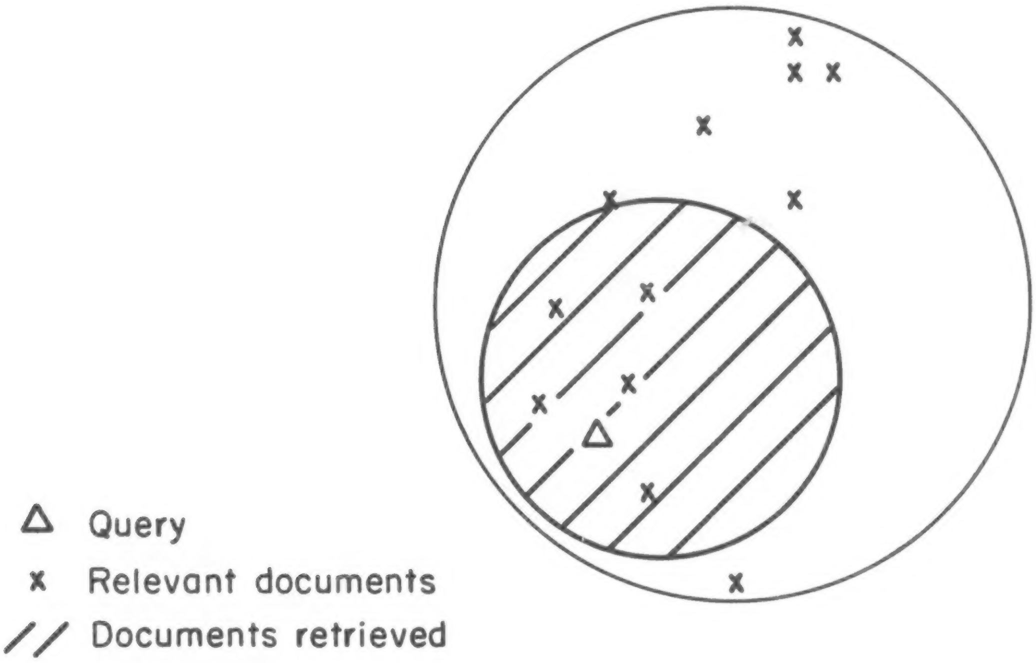
In connection with this approach, it is important to note that a large portion of normal queries covers more than one subject area. [7] In addition, concepts which appear in highly correlated nonrelevant may also be significant in retrieved relevant documents. As a result, since the basic selective negative feedback strategy of Johnson and Krablin leaves untouched those concepts in the query which may have been found in several of the highly correlated nonrelevant documents (and, as noted, several of the relevant retrieved as well), the query appears to remain in approximately the same area of the document space, as seen in Fig. 1. The highly correlated nonrelevant documents in the area may no longer be retrieved but the query also does not approach the documents relating to any secondary relevant subject area. The retrieval results confirm that most of the improvement obtained is caused by raising the ranks of the relevant documents in the primary subject area, and, in some cases, retrieving several other relevant in the same part of the document space.

In contrast, by removing those concepts in the query which are shown to be significant in the highly ranked nonrelevant documents, the query is moved from that part of the document space in which those documents appear, i.e. from an area of the space which is, in a sense, "more" nonrelevant than relevant to the query. It is hypothesized, as shown in Fig. 1, that the query is moved nearer to the set of documents related to its second subject area since presumably, the concepts which remain in the query relate to this area and, by removing the other concepts (or decreasing their weights), the remaining (or more weighty) concepts now assume primary importance in the query. In fact, a situation analagous to query splitting



a) Typical SMART Retrieval

b) Typical SMART Retrieval with relevance feedback to modify query


△ Query
x Relevant documents
// Documents retrieved

c) Typical retrieval with query modified by selective negative feedback (Methods 1,4)

Selective Negative Feedback Illustration

Fig. 1

is achieved, although relevant documents in the original area of the document space may now be overlooked. However, while missing these relevant documents, experiments show that the query is moved significantly nearer the second subject area and more new documents in this area are retrieved than would be the case if additional documents in the first subject area were retrieved by not modifying those selected concepts which appear in the query.

3. Selective Negative Relevance Feedback Strategies

The following procedure is used in testing the various selective negative feedback methods to be described.

1. A full search is made with the original queries (Note: As mentioned above, only those queries which retrieve no relevant in the first 5 documents retrieved are used in this study.)
2. Modify the query in one of the following ways (as summarized in Table 1):

   Method 1: Any concept which appears in at least 3 of the first 5 nonrelevant documents is deleted it it appears in the query. No new concepts are added to the query.

   Method 2: Any concept which appears in at at least 3 of the first 5 nonrelevant documents is assigned a weight equal to the average of its weights in these documents multiplied by -1. If the concept appears in the query, its weight is replaced by the new calculated weight. If the concept does not appear in the query, it is added to the query.

Method 3: This method is similar to Method 2 but if the selected concept appears in the query, the new (negative) weight of the concept is added to its present weight in the query.

Method 4: This method is similar to Method 1 but a concept must appear in all 5 nonrelevant documents in order to be selected.

Method 5: This technique is similar to Method 2 but a concept must appear in all 5 nonrelevant documents to be selected.

3. Search the document collection with the modified query, and repeat procedure of part 2.

This process is halted when a satisfactory proportion of relevant documents are retrieved.

For comparison, searches are made with the test queries using a standard method of negative relevance feedback in which the nonrelevant document retrieved with rank 1 in the original search is subtracted from the query and a subsequent search is made with the modified query. Two feedback iterations are performed.

In Methods 1 and 4, the danger exists of reducing the query to the zero vector. It has been found that such reduction occurs after the second iteration of Method 1 with only 2 queries. However, the experiments performed indicate that two iterations are the maximum number desirable, as further iterations cause too much distortion in the query.

4. The Experimental Environment

The strategies outlined above have been tested on the Cranfield collection of 424 document vector abstracts produced using a word form the-

Method 1:	Any concept which appears in at least 3 of the 5 nonrelevant documents is deleted if it appears in the query. No new concepts are added to the query.
Method 2:	Any concept which appears in at least 3 of the 5 nonrelevant documents is assigned a weight equal to the average of its weight in the 5 documents multiplied by -1. If the concept appears in the query, its weight is replaced by the calculated weight. If the concept does not appear in the query, it is added to the query.
Method 3:	This method is similar to Method 2 but if the selected concept appears in the query, the calculated (negative) weight of the concept is added to its present weight in the query.
Method 4:	This method is similar to Method 1 but a concept must appear in all of the 5 nonrelevant documents in order to be selected.
Method 5:	This method is similar to Method 3 but a concept must appear in all 5 of the nonrelevant documents to be selected.

Five Proposed Selective Negative Feedback Schemes

Table 1

saurus and 155 queries, 35 of which retrieve no relevant in the first five documents retrieved. These queries are used as the experimental base. In the experiment, 15 documents are shown to the user but only the first five are used for relevance feedback.

5. Experimental Results

Since it is hypothesized that modification of the query by the proposed methods moves it to a part of the document space which represents the second subject area, it is important to consider the number of new relevant documents which are retrieved in the first 15 documents, i. e. those which have not previously been shown to the user. [7,8] As seen in Table 2, Method 1 is the most successful in retrieving new relevant documents. In one iteration 24 relevant documents appear in the first 15 documents retrieved or 15.5% of the remaining relevant documents, with an average of 3.0 concepts deleted from each query. In two iterations, a total of 30 new relevant documents are shown to the user or 19.4% of the remaining relevant in the collection for this particular set of queries. Method 4, which requires that a concept appear in all 5 nonrelevant documents in order to be deleted, retrieves 16 new documents or 10.3% of the remaining relevant, with an average of 1.6 concepts deleted from each query. The techniques which add concepts with negative weights to the query show inferior results. Method 2 retrieves only 9 new documents or 5.8% of the remaining relevant while Method 3 retrieves 8 new relevant documents. Thus it appears that assigning a weight of zero to a concept, i. e., deleting it from the query, results in less distortion of the query than assigning it a negative weight. In addition, Methods 1 and 4, which both neglect to add new concepts with nega-

	Method 1 (1 iter.)	Method 1 (2 iters.)	Method 2	Method 3	Method 4	Method 5
Number of queries modified	34	34	34	34	22	22
Number of relevant in first 5 retrieved	13	14	8	9	12	10
Number of relevant in first 15 retrieved	38	28	10	10	33	21
Number of new relevant in first 5 retrieved	13	16	8	9	11	9
Number of new relevant in first 15 retrieved	24	30	9	9	16	13
% of remaining relevant retrieved in first 15	15.5	19.4	5.8	5.8	10.3	8.4
Number of queries which retrieve at least 1 new relevant in the first 15	18	24	9	9	13	11

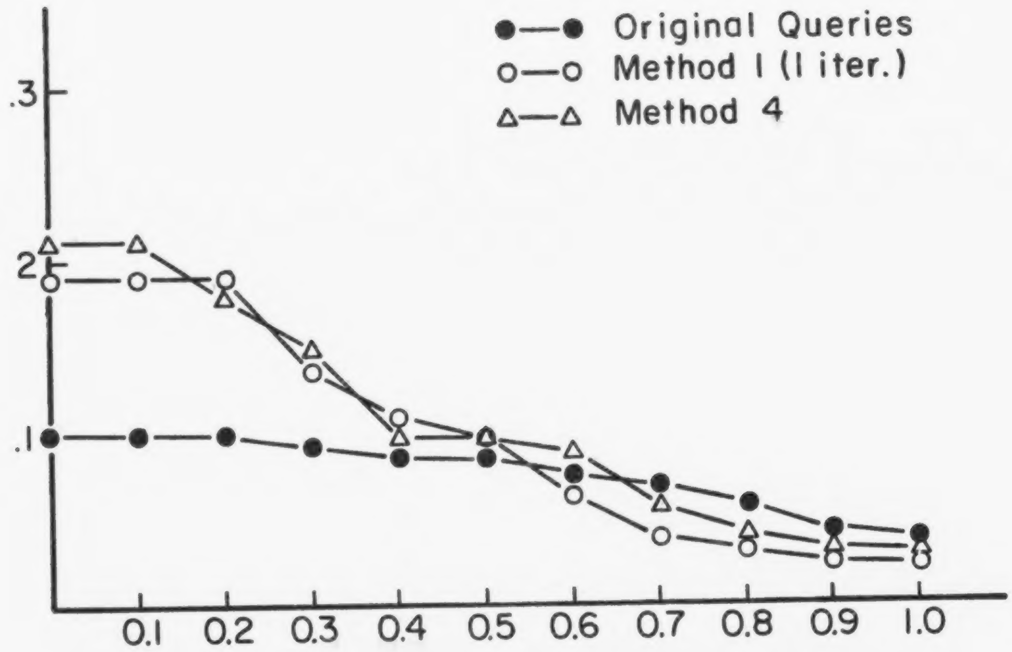
Comparison of Methods 1-5

Table 2

tive weights to the query, are significantly more successful in retrieving new relevant documents than Methods 2, 3, and 5, which do add new concepts with negative weights.

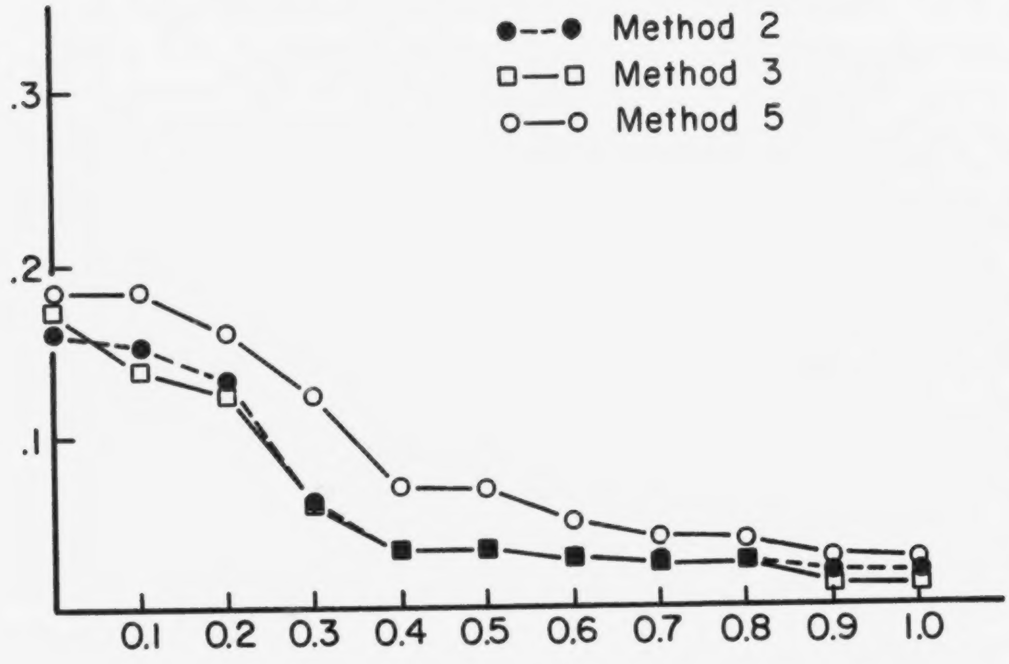
As seen in Fig. 2, the more selective modification technique of Method 4 results in higher precision figures at recall levels up to 0.5 than those achieved by Method 1, although precision figures for Method 1 are higher at the higher recall levels. It is also seen by examination of retrieval results that in some cases for Method 1, the ranks of relevant documents which are retrieved among the top 15 documents in the original search decrease significantly since, as hypothesized, the query is moving in a direction away from these highly correlated documents. As shown in Table 2, for Method 1, 24 of the 38 relevant documents retrieved, or 63%, are new relevant documents. Since Method 4 leaves 13 queries unchanged, the high ranks of these relevant documents remain the same and thus help in achieving high precision figures for Method 4 at low recall levels. In the same way, Method 1 tends to push low ranking relevant documents lower if these documents are in the area of the document space from which the query is being moved, as they tend to be. In fact, using Method 1, 47 relevant documents which have a nonzero correlation with the queries are reduced to having a zero correlation with the modified queries after one iteration. It is to be noted that some of these relevant documents have been seen by the user, as they appear in the top 15 retrieved documents, but, nonetheless, such factors affect the precision and recall calculations.

As seen in Table 3, the standard feedback technique of subtracting the nonrelevant document with rank 1 from the query only retrieves 13 new relevant documents after 2 iterations, or 8.4% of the remaining relevant.



Recall-Precision Curve for Original Queries, Methods I and 4

Fig. 2



Recall-Precision Curve for Methods 2,3 and 5

Fig. 3

	Iteration 1	Iteration 2	Combined
Number of relevant in first 5 retrieved	3	3	4
Number of relevant in first 15 retrieved	9	12	17
Number of new relevant in first 5 retrieved	3	1	4
Number of new relevant in first 15 retrieved	8	5	13
% of remaining relevant retrieved in first 15	5.2	3.2	8.4
Number of queries which retrieve at least 1 new relevant in the first 15	5	4	9
Average number of concepts subtracted from the query	56.2	35.9	92.1

Results for Nonselective Negative Feedback Scheme

Table 3

The high average number of concepts subtracted from the query after two iterations, 92.1, may explain the poor performance as the query is probably overperturbed.

6. Evaluation of Experimental Results

As the criteria cited above (number of new relevant retrieved, etc.) as well as the statistical T- and Wilcoxon Signed Rank tests favor Methods 1 and 4 significantly over Methods 2, 3, and 5, only the former are compared with the standard nonselective negative feedback scheme and with each other.

According to the T-test, the differences in performance between Method 1 and Method 4 are statistically significant. Using measures of rank recall, log precision, normalized recall, normalized precision, and recall level averages, Method 4 is concluded to be "better" than Method 1. The Wilcoxon Signed Rank test confirms this conclusion.

The Sign test favors the nonselective negative feedback strategy over Method 1 while the same test favors Method 4 over nonselective negative feedback. However, as noted above and by others, [7,8] several other factors must be considered in evaluating the various strategies.

Methods 1 and 4 both perform better than the nonselective negative feedback scheme as reflected by the number of new relevant retrieved. This is also reflected in the standard precision-recall curves (see Tables 2 and 3, Figs. 1 and 3). As noted previously, the improved precision-recall curves for these methods do not result from simply raising the ranks of already retrieved relevant for, as shown in Table 2, 63% of the relevant documents retrieved by Method 1 are new documents not seen before by the user. For Method 4, 48% of the relevant documents retrieved are new.

To determine which of Methods 1 or 4 is to be favored, it must be considered that although the precision-recall curve of Method 4 is higher than that of Method 1 at recall levels up to 0.5, the curve for Method 1 shows higher precision at recall levels greater than 0.5, since more relevant are retrieved using Method 1 than if Method 4 is used. At low recall levels, precision may be improved by raising the ranks of relevant documents already shown to the user. As noted by Hall et al [7] and Cirillo et al [8], assuming that 15 documents are shown to the user, whether a relevant document is ranked 8 or 13 is not important to the user since he is shown both documents; it is in the higher ranks of relevant documents retrieved that Method 4 seems to show better performance figures than Method 1.

It is, in addition, important to note that Method 4, due to its more selective modification procedure which requires that a concept appear in all 5 nonrelevant documents in order to be deleted from the query, fails to alter 13 of the 35 queries while Method 1 modifies 34 of the 35 queries. For those queries which are modified, their performances as far as the number of new relevant documents retrieved are similar. Method 1 retrieves an average of .71 new documents per query and Method 4 retrieves an average of .73 new documents per query.

Since negative feedback schemes are conceived for the purpose of dealing with problem queries, i.e. those which retrieve no relevant in the first 5 documents retrieved, and thus cannot be modified by positive feedback schemes employing relevant documents, a strategy which leaves 37% of the queries unmodified must be considered unsatisfactory for the purpose for which it is designed.

Therefore, it is recommended that Method 4, which deletes from the query those concepts which appear in at least 3 of the 5 nonrelevant documents, be used as a negative feedback scheme for those queries which retrieve no relevant documents in the first 5 retrieved. However, as it is hypothesized in the present study that the large number of new relevant documents retrieved by queries modified by this strategy are obtained by moving the query to a new section of the document space, which represents its second subject area, it is necessary to perform further experiments to determine how to retrieve the relevant which remain unretrieved in that part of the document space which relates to its first subject area. A combination of such techniques would presumably result in significantly better retrieval results for the problem queries dealt with in this study.

Rank	Initial		Nonselective		Method 1		Method 4	
	Doc	Corr	Doc	Corr	Doc	Corr	Doc	Corr
1	106	.4471	372	.0261	226R	.3993	226R	.3458
2	87	.4429	321	.0148	340	.3592	340	.3111
3	74	.4248	373	.0097	227R	.3118	225R	.2725
4	27	.4025	103	.0	238	.3093	321	.2722
5	128	.3643	197	.0	244	.3020	227R	.2700
6	91	.3593	241	.0	267	.2993	238	.2679
7	72	.3542	264	.0	167	.2867	244	.2616
8	83	.3539	267	.0	372	.2774	267	.2592
9	387	.3501	273	.0	225R	.2697	167	.2483
10	107	.3476	320	.0	339	.2649	372	.2402
11	167	.3441	106	-.9917	321	.2357	339	.2294
12	234	.3274	107	-.5190	270	.2200	270	.2223
13	225R	.3237	91	-.4715	374	.2025	228R	.1816
14	62	.3227	36	-.4627	243	.1992	374	.1754
15	65	.3227	415	-.4446	242	.1911	243	.1725

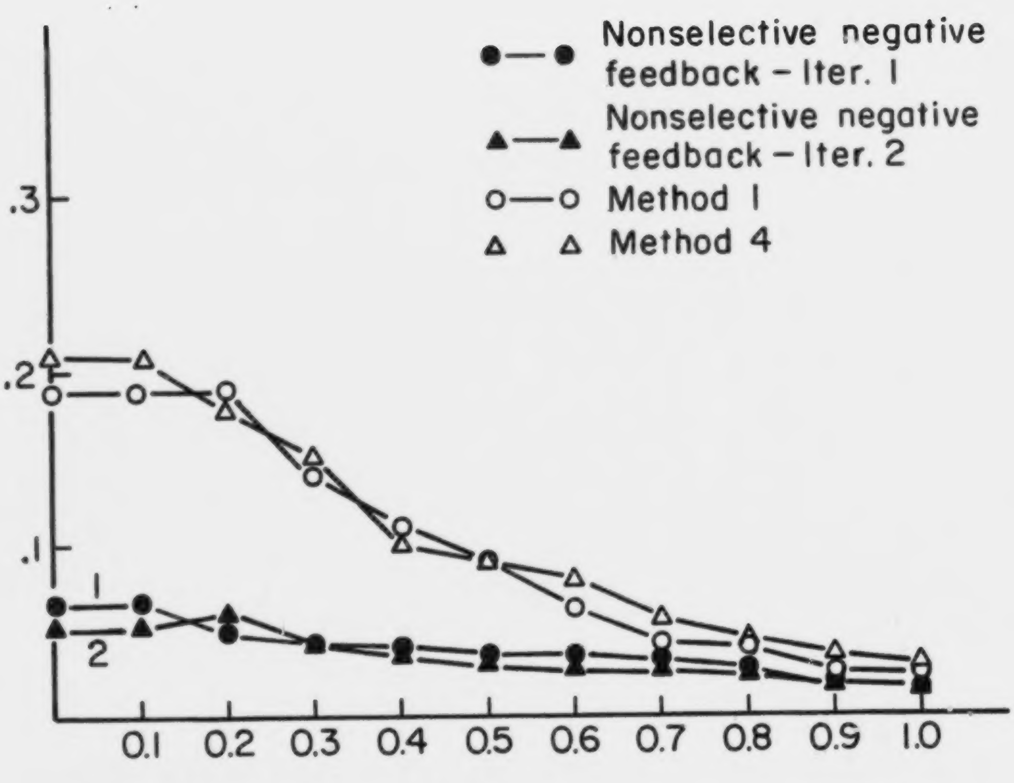
a) Three Negative Feedback Strategies for Query 34

Rank	Doc	Corr	Doc	Corr	Doc	Corr	Doc	Corr
1	73	.3230	73	-.9971	73	.3322	163R	.2011
2	406	.2926	174	-.4847	406	.3084	202	.1964
3	40	.2363	133	-.4199	40	.2491	413	.1474
4	367	.2349	134	-.4158	174	.2430	385	.1367
5	398	.2333	398	-.4071	74	.2185	203	.1297
6	174	.2305	406	-.4054	7	.2148	384R	.1235
7	381	.2173	419R	-.4032	367	.2063	61	.1223
8	74	.2073	234	-.3997	90R	.2010	90R	.1066
9	7	.2038	381	-.3737	234	.1991	73	.1057
10	163R	.1962	28R	-.3733	398	.1933	122	.1003
11	90R	.1907	136	-.3674	394	.1907	26	.0898
12	234	.1889	40	-.3593	202	.1852	70	.0898
13	394	.1809	7	-.3513	65	.1811	22	.0884
14	202	.1757	74	-.3486	64	.1794	39	.0881
15	65	.1718	376	-.3485	163R	.1724	7	.0876

b) Three Negative Feedback Strategies for Query 137

Retrieval Results for Three Negative Feedback Strategies

Table 4



Recall-Precision Curve for Nonselective Negative Feedback Technique, Methods 1 and 4.

Fig. 4

Query	Orig. Search	Method 1 1st iter	New Rel	Method 2 2nd iter	New Rel	Total New	Method 2	New Rel	Method 3	New Rel
3	3	3	0	3	0	0	1	0	1	0
6	1	2	1	2	0	1	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	1	1	1
12	1	4	3	5	1	4	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0
29	1	1	0	0	0	0	0	0	0	0
32	0	4	4	0	0	4	1	1	1	1
33	1	1	1	2	0	1	0	0	0	0
34	1	3	2	0	0	2	1	1	1	1
37	1	1	0	0	0	0	0	0	0	0
38	1	0	0	0	0	0	0	0	0	0
45	1	2	1	0	0	1	0	0	0	0
46	2	4	3	3	1	4	1	1	1	1
48	0	1	1	1	0	1	1	1	1	1
54	0	0	0	1	1	1	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0	0
58	1	0	0	0	0	0	0	0	0	0
66	1	0	0	1	1	1	0	0	0	0
74	2	1	1	1	0	1	0	0	0	0
76	1	2	1	1	0	1	1	1	1	1
78	0	0	0	0	0	0	0	0	0	0
79	1	1	1	0	0	1	0	0	0	0
83	1	0	0	0	0	0	0	0	0	0
91	2	1	1	0	0	1	0	0	0	0
95	0	0	0	1	1	1	2	2	2	2
96	0	1	1	2	1	2	0	0	0	0
102	0	1	1	0	0	1	0	0	0	0
103	0	0	0	0	0	0	0	0	0	0
113	2	1	0	0	0	0	0	0	0	0
118	0	2	2	2	0	2	0	0	0	0
134	1	0	0	1	0	0	0	0	0	0
137	2	2	0	2	0	0	1	1	1	1
140	1	0	0	0	0	0	0	0	0	0

Number of Relevant in First 15 Retrieved for Various Feedback Methods

Table 5

Query	Method 4	New Rel	Method 5	New Rel	Nonsel 1st iter	New Rel	Nonsel 2nd iter	New Rel	Total New
3	3	0	0	0	0	0	0	0	0
6	2	1	2	1	0	0	1	1	1
8	0	0	0	0	0	0	0	0	0
10	0	0	0	0	1	1	0	0	1
12	1	0	2	1	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0
29	1	0	1	0	1	1	2	1	2
32	0	0	0	0	2	2	1	0	2
33	1	0	1	0	0	0	0	0	0
34	4	3	2	2	0	0	0	0	0
37	1	0	1	0	0	0	0	0	0
38	1	0	1	0	0	0	0	0	0
45	0	0	0	0	0	0	2	2	2
46	4	3	2	2	0	0	0	0	0
48	1	1	1	1	0	0	0	0	0
54	0	0	0	0	0	0	0	0	0
55	1	1	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	0
66	1	0	1	0	3	2	0	0	2
74	1	1	1	1	0	0	0	0	0
76	2	1	1	1	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0
79	1	0	1	0	0	0	0	0	0
83	0	0	0	0	0	0	0	0	0
91	1	1	0	0	0	0	0	0	0
95	0	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0	0
102	0	0	0	0	0	0	1	1	1
103	0	0	1	1	0	0	1	1	1
113	1	0	0	0	0	0	1	0	0
118	1	1	0	0	0	0	0	0	0
134	1	0	1	0	0	0	0	0	0
137	3	2	1	1	2	2	2	0	2
140	1	1	1	1	0	0	1	1	1

Number of Relevant in First 15 Retrieved for Various Feedback Methods

Table 5 (contd.)

References

[1] J. Kelly, Negative Response Relevance Feedback, Information Storage and Retrieval, Scientific Report ISR-12 to the National Science Foundation, Section IX, Department of Computer Science, Cornell University, June 1967.

[2] E. Ide and G. Salton, Interactive Search Strategies and Dynamic File Organization in Information Retrieval, Information Storage and Retrieval, Scientific Report ISR-16 to the National Science Foundation, Section XI, Department of Computer Science, Cornell University, September 1969.

[3] G. Salton, Automatic Information Organization and Retrieval, McGraw Hill, Inc., New York 1968.

[4] W. Riddle, T. Horwitz, and R. Dietz, Relevance Feedback in an Information Retrieval System, Information Storage and Retrieval, Scientific Report ISR-11 to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, June 1966.

[5] E. Ide, New Experiments in Relevance Feedback, Information Storage and Retrieval, Scientific Report ISR-14 to the National Science Foundation, Section VIII, Department of Computer Science, Cornell University, September 1968.

[6] P. Johnson and L. Krablin, Proposal for Modified Nonrelevant Feedback, Computer Science 635 Term Project Report, June 1969.

[7] H. Hall and N. Weiderman, The Evaluation Problem in Relevance Feedback Systems, Scientific Report ISR-12 to the National Science Foundation, Section XII, Department of Computer Science, Cornell University, June 1967.

[8] C. Cirillo, Y. K. Chang, and J. Razon, Evaluation of Feedback Retrieval using Modified Freezing, Residual Collection and Test and Control Groups, Scientific Report ISR-16 to the National Science Foundation, Section X, Department of Computer Science, Cornell University, September 1969.

XI. The Use of Past Relevance Decisions in Relevance Feedback

L. Paavola

Abstract

A high degree of similarity may be expected to exist among documents judged to be relevant to the same query. This paper investigates some possibilities for exploiting this potential similarity in relevance feedback. Runs are made on the ADI and Cranfield 424 collections of the SMART retrieval system. In these runs all "jointly relevant" documents are incorporated into feedback as if they were a single relevant document. Standard recall-precision evaluation measures are used, and the performance of some individual queries is illustrated. Some directions for further research are suggested.

1. Introduction

In the SMART system, statistical and syntactic analyses of search queries and documents are used for text analysis, and automatic comparisons of analyzed queries to documents or to sets of centroids of document clusters are used for the selection of documents to be displayed to query authors. [1] However, the utility of these methods alone is severely limited, and attempts have been made to introduce subjective judgments into the retrieval process. The usual method, known as relevance feedback, uses a query author's decisions about the relevance to his query of specified documents in order to modify the vector representation of his query. [1, section 7-4] Occasionally such judgments are used to modify document vectors. [2] Methods which do not alter query or document vectors include query splitting [3] and query clustering. [4,5]

2. Assumptions and Hypotheses

This paper details another method of using the history of a system to improve its performance. The assumption is made that if a given document is known to be relevant to a query, another document is more likely to be relevant to the query if both have been judged relevant to some past query. It is further assumed that the number of such past occurrences of joint relevance may be a useful index to inter-document similarity.

The following problems may be anticipated in such a system: the system may be handicapped in dealing with queries of a type which it has not encountered frequently earlier; user ideas of relevance and nonrelevance may differ widely; unless special measures are taken, documents which may be relevant to a given query but never initially retrieved (e.g. situations in which query splitting would be in order) may become increasingly less likely ever to be retrieved.

The proposed method is expected to have the following advantages: general queries with a high number of relevant documents may establish a loose connection between documents of the same general subject area, while specific queries may set up stronger connections between more closely similar documents; the system may function well for the "average" user, if queries do not vary too widely; groups of documents of which all are relevant to each of several queries may be used to better performance.

3. Experimental Method

The procedure is first tried on the ADI collection of the SMART retrieval system, consisting of 82 documents and 35 queries, then on the Cranfield 424 collection, which has 424 documents and 155 queries. In each case, the query collection is divided into two equal groups by random methods. The documents relevant to each query are known. From the relevance decisions for the queries in the first group a list is made for each document of the other documents with which it has been included in such decisions and the number of times for each, as shown in Fig. 1.

The other half of the query collection is used to make three searches of the entire document collection. The first search is a full search using unaltered query vectors. The second search incorporates in positive feedback those documents among the first five shown the user which are judged relevant by him. The third search alters the query vectors in the way described below.

In general, the altered query is constructed according to the following formula:

$$q = a_0 q_0 + a_1 \left(\sum_{i=1}^{N_R} D_{R_i} \right) + a_2 \left(\sum_{i=1}^{N_{JR}} (a_3 n_{D_i} + a_4 n_{J_i}) \, D_{JR_i} \right) ,$$

where q = the altered query

Documents 1, 3, 35, 36, 89	are relevant to query 1
Documents 2, 8, 35, 36, 89, 90	are relevant to query 2
Documents 4, 36, 90	are relevant to query 3

Document	J-r docs*	#**
1	3	1
	35	1
	36	1
	89	1
2	8	1
	35	1
	36	1
	89	1
	90	1
3	1	1
	35	1
	36	1
	89	1
4	36	1
	90	1
8	2	1
	35	1
	36	1
	89	1
	90	1
35	1	1
	2	1
	3	1
	8	1
	36	2
	89	2
	90	1

Document	J-r docs*	#**
36	1	1
	2	1
	3	1
	4	1
	8	1
	35	2
	89	2
	90	2
89	1	1
	2	1
	3	1
	8	1
	35	2
	36	2
	90	1
90	2	1
	4	1
	8	1
	35	1
	36	2
	89	1

* Documents joint-relevant to the given document

** Number of times each joint-relevant document occurs in a list of relevant documents with the given document

Examples of Joint Relevance

Fig. 1

q_0 = the initial query

D_{R_i} = the relevant documents among the top n (here n=5), according to the ranking produced by the full search

N_R = the number of such documents D_{R_i}

D_{JR_i} = documents joint relevant to any of the D_{R_i}

N_{JR} = the number of such documents D_{JR_i}

n_{D_i} = the number of D_{R_i} to which a particular D_{JR_i} has been found to be joint relevant

n_{J_i} = total number of joint relevancy decisions of the particular D_{JR_i} with any of the D_{R_i}

a_0 =
a_1 =
a_2 = adjustable parameters
a_3 =
a_4 =

(One may choose to include in feedback only those D_{JR_i} which have n_{J_i} greater than a certain minimum value.)

An example of the use of this notation is given in Fig. 2. The particular coefficients that have been tried for the Cranfield 424 collection are $a_0 = 100$, $a_1 = 100$, $a_2 = 100/(\sum_{i=1}^{N_{JR}} n_{D_i})$, $a_3 = 0$, and $a_4 = 1$. Parameter a_2 is normalized because some documents

Documents 5, 6, 89, 312, and 400 shown to user.

He identifies 6, 89, and 400 as relevant.

Joint relevance lists for these documents:

6		89		400	
32	3	51	1	5	2
51	3	71	1	89	1
65	1	212	1	93	1
212	1	400	1	284	1
312	2				
400	2				

$D_{R_1} = 6$, $D_{R_2} = 89$, $D_{R_3} = 400$; $N_R = 3$; $N_{JR} = 10$

i	D_{JR_i}	n_{D_i}	n_{J_i}
1	5	1	2
2	32	1	3
3	51	2	4
4	65	1	1
5	71	1	1
6	93	1	1
7	212	2	2
8	284	1	1
9	312	1	2
10	400	2	3

Computation of Joint Relevance Parameters

Fig. 2

in the collection have an extremely large number of joint-relevant documents, while others have none.

The successive definitions of the query of Fig. 2 are illustrated in Fig. 3. The query used for the pure positive feedback search and that used for the joint relevance search are always identical except that in the latter certain concepts have increased weight and other concepts are added.

To obtain a final evaluation, the simple feedback and joint relevance runs are compared to each other (and to the full search) by the AVERAGE and VERIFY routines.

4. Evaluation

The run on the ADI collection shows enough difference between the two methods to merit a run on the Cranfield collection. The chi square probabilities were 0.0001 for the t-test; 0.0483 for the sign test without ties, 1.0000 with ties; and 0.0006 for the Wilcoxon test.

Recall-precision for the Cranfield 424 collection are displayed in Fig. 4. The higher precision at low recall for simple positive feedback is probably due to the inability of a vector loaded with many concepts to be very accurate in choosing the highest-ranking documents, although performing well on the whole. From the graph of Fig. 4, it is seen that the simple feedback method is more advisable than the particular joint relevance strategy tried when only the ranking of the documents at the top is important.

Of the relevant documents which were changed in rank by

Search 1

$$q = q_0$$

Search 2

$$q = q_0 + 1(6) + 1(89) + 1(400)$$

Search 3

$$\begin{aligned} q = 100q_0 &+ 100(6) + 100(89) + 100(400) \\ &+ \frac{100}{25}\,(2(5) + 3(32) + 4(51) + 1(65) \\ &+ 1(71) + 1(93) + 2(212) + 1(284) \\ &+ 2(312) + 3(400)) \end{aligned}$$

3(400), e.g., means document 400 is added in with weight 3.

Query Alteration

Fig. 3

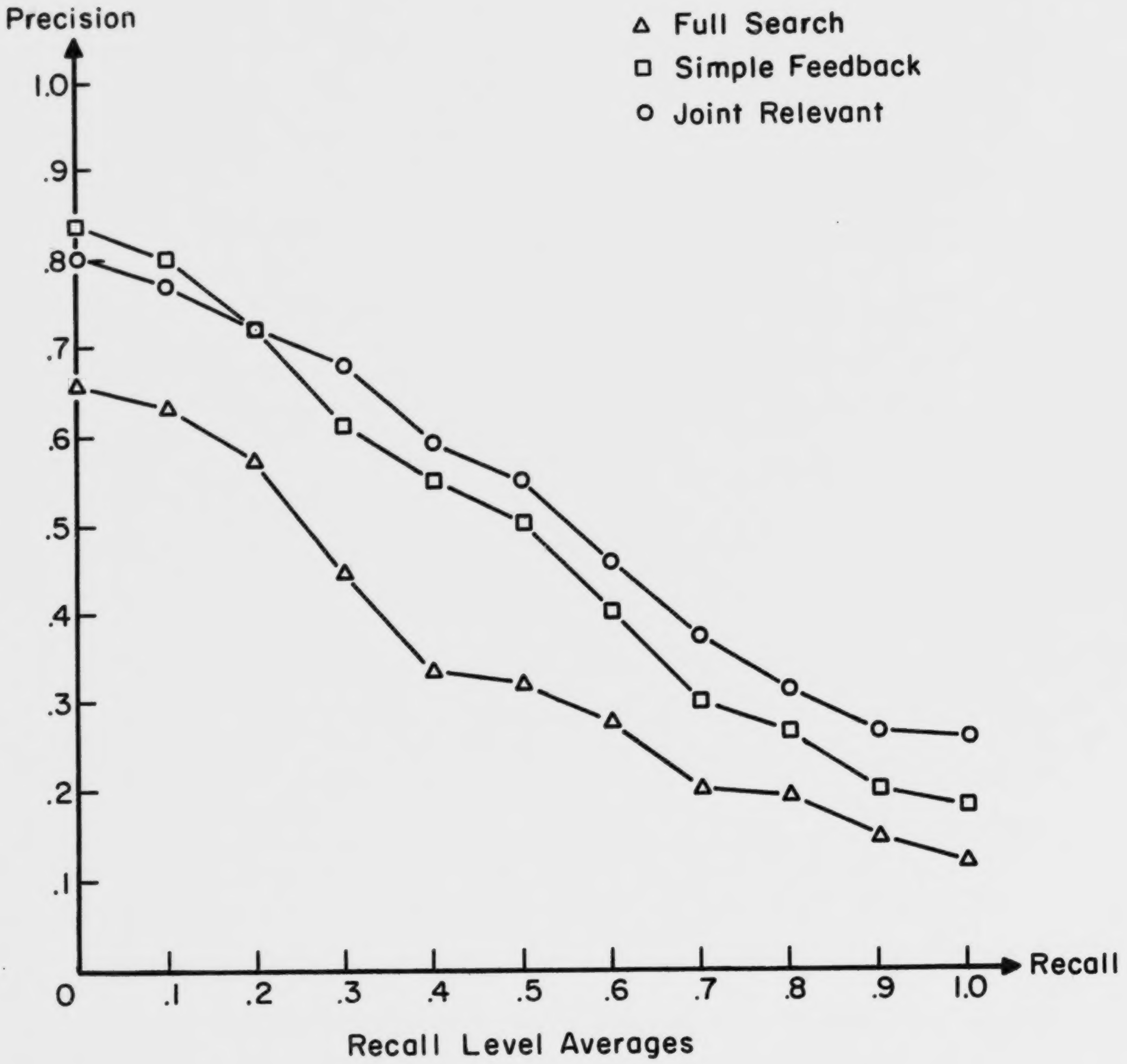

Precision
1.0
.9
.8
.7
.6
.5
.4
.3
.2
.1
0
.1
.2
.3
.4
.5
.6
.7
.8
.9
1.0
Recall
Recall Level Averages
△ Full Search
□ Simple Feedback
○ Joint Relevant

Fig. 4

the joint relevance process, 59 obtain lower ranks and 127 receive higher ranks. The probability under the null hypothesis of a chi square larger than that observed is 0.0000 for the t-test, Wilcoxon test, and sign test without ties; for the sign test using ties the probability is 1.0000. The large number of ties can be attributed to the lack of joint relevance information to be added into many of the queries.

Performance of the simple feedback and joint relevance searches are shown for several queries in Fig. 5. Sometimes the addition of joint relevance information does not substantially affect the effectiveness of the query one way or the other (e.g. query 54). Sometimes it actually moves the query away from relevant documents (query 26). But often it produces dramatic improvement (query 77). Sometimes the improvement is due to the direct addition of relevant documents (query 13), some of which would have been more effective had they had greater weight. Sometimes very few relevant documents are added, but the important concepts are nevertheless amplified by inclusion of joint relevance information (query 42). Sometimes the inclusion of both produces improvement (query 7). Sometimes the additions dilute the query (query 61).

The above analysis supports the conclusion that inclusion of joint relevance information, even if restricted to the weight of one relevant document only, produces significant improvement.

In evaluating this experiment one must keep in mind the differences between the experimental situation and an actual one. The results are biased positively by the fact that in an actual

Query 7

R	F	J
1	54R	54R
15	173	7R
41	7R	
64		123R
70		56R
91	123R	
102	56R	
107		121R
132		126R
137	126R	
138		124R
147		122R
160	121R	120R
168		125R
219	124R	
231	120R	
233	122R	
249	125R	

Query 13

R	F	J
1	72R	72R
2	62R	62R
3	83	5R
4	91	6R
14	5R	234
54	5R	

Query 26

R	F	J
1	58R	58R
3	378R	169
11	184	91R
33	34R	
42	91R	
49		378R
57	151R	
58		34R
96	78R	
98		151R
145		78R

Query 42

R	F	J
1	294R	294R
3	80	293R
5	296	292R
7	287	290R
9	293R	291R
11	7R	289R
20	94	43R
39	292R	
50	43R	
104	289R	
120	290R	
235	291R	

Query 54

R	F	J
1	124R	124R
2	125R	125R
8	55R	55R
13	411	136R
14	392	173R
21	82R	351
23	79	82R
28	30R	112
29	388R	61
37	173R	
38	136R	
39	10R	
48		10R
49		100R
50	101R	
54		30R
55	100R	101R
60		388R
61		164R
76		184R
80	164R	
81	38R	
86	118R	
87		391R
88	391R	
102	364R	
106	150R	
116	384R	
118		364R
128		38R
142		118R
150		150R

Query 61

R	F	J
1	141R	142R
2	142R	141R
3	143R	143R
5	145R	145R
6	221R	144
7	140	221R

Query 71

R	F	J
1	89R	418R
2	344	89R
3	264	420R
5	418R	268
38	420R	

Query 7: 9(7), 5(11), 5(40), 9(48), 100(54), 9(56), 9(98), 9(120), 5(121), 9(122), 9(123), 9(124), 9(125), 5(126)

Query 13: 33(5), 33(6), 33(62), 100(72)

Query 26: 33(7), 100(58), 33(80), 33(169)

Query 42: 33(92), 33(293), 100(294), 33(307)

Query 54: 8(7), 4(48), 8(54), 4(55), 8(56), 4(59), 4(98), 12(120), 4(121), 8(122), 8(123), 100(124), 8(125), 4(126), 4(308), 4(310), 4(311)

Query 61: 100(141), 100(142), 100(143), 100(144), 100(145)

Query 71: 100(401)

Key:

R = Rank

F = Simple Feedback Search

J = Joint Relevance Search

The D_{JR_i} are given in the following form:

33(292) implies that document 292 was added in with weight 33

Search Results for Joint Relevance Modification

Fig. 5

application not all the documents relevant to a given query would be shown to a user, identified as relevant, and added to the joint relevance lists. They are biased negatively by the fact that relevance information obtained by a query in the second half of the collection might often help a new query subsequently submitted in the second half; this effect could not be taken into account in the experimental design. A sounder though more laborious experiment would have been to run the entire query collection against the document collection, while updating the joint relevance lists after each query. Still more significant results would have been obtained had the joint relevance lists been composed of only those documents which a user might see and identify as relevant. However, such experiments are difficult to perform without adequate system support.

5. Conclusions

The assumptions of part 2 are found to be largely justifiable, although the importance of the number of past joint relevance decisions should be further investigated. The danger of biasing the system toward one type of query is avoided, since the two halves of the query collection are fairly similar. The experiments are not extensive enough to detect isolation of documents. As expected, loose and strong connections are established by general and specific queries, respectively. The joint relevance procedure does take advantage of document groups. And a partial but important answer to the weighting problem is that greater emphasis should be placed on the joint relevant documents, although ways must be found to counteract the negative effects of such an increase. Perhaps this effect may be partially counteracted as the number of queries run through a system increases.

In a new experiment, low-weight concepts might be eliminated from altered queries. Certainly better values for a_0, a_1, a_2, a_3, and a_4 should be found. There may be possibilities for the use of joint-relevance information in negative feedback. Incorporation of the best known feedback strategies into the joint-relevance query alteration equation should be attempted. Perhaps high-frequency occurrence in joint-relevance lists of a document already known to be relevant should lead to a higher weighting of such a document.

The experimental data indicate that the use of joint relevance information is a valuable tool in information retrieval, that more testing of procedures for using this information is in order, and that the nature of the tradeoff between computational complexity and effectiveness of additional information must be determined for such procedures.

References

[1] G. Salton, Automatic Information Organization and Retrieval, McGraw Hill, Inc., New York 1968.

[2] T. L. Brauen, Document Vector Modification in On-Line Information Retrieval Systems, Information Storage and Retrieval, Report ISR-17 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[3] A. Borodin, L. Kerr, F. Lewis, Query Splitting in Relevance Feedback Systems, Information Storage and Retrieval, Report ISR-14 to the National Science Foundation, Section XII, Department of Computer Science, Cornell University, October 1968.

[4] V. R. Lesser, A Modified Two-Level Search Algorithm Using Request Clustering, Information Storage and Retrieval, Report ISR-11 to the National Science Foundation, Section VII, Departmentof Computer Science, Cornell University, June 1966.

[5] S. Worona, Query Clustering in a Large Document Space, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section XV, Department of Computer Science, Cornell University, September 1969.

XII. A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering

D. B. Johnson and J. M. Lafuente

Abstract

A single pass clustering method is presented, which compares favorably with more expensive clustering algoritms. During the clustering process various parameters are controlled, such as number of clusters, size of the clusters and amount of overlap. The method is tested using the ADI collection of 82 documents and the Cranfield 424 collection. The results are compared to full search and to results obtained by searching clusters produced by Dattola's algorithm. The effect of ordering of the collection is investigated and some variation is obtained in the results for different orderings. Single-level as well as two-level clustering is considered. The results, in general, point to better performance with multilevel clustering and some suggestions for extending the algorithm to include multilevel clustering are given.

1. Introduction

An important consideration in an automatic information retrieval system is the time spent in searching a collection. To avoid searching the entire collection, it becomes necessary to classify documents into related groups. This is the technique of clustering. Documents are grouped into clusters by assigning items containing similar concepts to the same cluster. A centroid vector is constructed for each cluster, and queries are matched at first against these centroids. Only those clusters comparing favorably with a query are then searched in the normal manner. Thus a sacrifice in time to produce the clusters is compensated by later savings in the search time. The problem is how to develop efficient techniques for producing meaningful clusters in order to minimize searching costs. The suitability of any clustering method must then be measured according to the following criteria:

1. Cost of generating the clusters;
2. Cost of searching the clustered collection;
3. Effectiveness of the search process, usually measured by evaluating recall and precision.

The clustering problem becomes critical with very large collections. Comparing each document with every other document is no longer feasible, and efficient algorithms have been developed which attempt to minimize the number of document comparisons. Even with these methods, the classification of a collection containing several thousand items is a time consuming process.

This paper describes a clustering algorithm which makes a single pass over a collection. Each document is examined only once and clusters are formed in the process. A document is considered for inclusion into one or more existing clusters before it is allowed to begin a cluster of its own. Various parameters such as cluster size, number of clusters, and amount of overlap are controlled throughout the clustering process. The algorithm is tested using the ADI collection of 82 documents and 35 queries available in the SMART system. The algorithm, however, is designed with a view toward large collections.

2. Methods of Clustering

Various methods have been devised for clustering. Usually these methods require the computation of a correlation matrix, representing the correlation of each document with every other document in the collection, followed by a grouping of those documents which correlate best with each other. Input parameters for these clustering algorithms include the number of clusters desired, the maximum size of each cluster, the amount of overlap and the number of "loose" or unclassified documents to be allowed.

Two clustering methods are presently in use in the SMART System; [1] they are: Rocchio's clustering algorithm [2] and a variation of Doyle's algorithm. [3] In Rocchio's algorithm, each unclustered document is selected as a candidate for a cluster nucleus. All remaining documents are then correlated with it, and the document is subjected to a density test based on cut-off correlation coefficients. If the document passes the density test, a new cluster is formed and a cut-off correlation is determined based on the relative distribution of correlations with the given document. A centroid vector is then computed by combining all concepts of those documents with correlation above the computed cut-off value. The centroid vector is next matched against the entire collection to create an altered cluster. The entire procedure is now repeated with all unclustered documents until all documents are either clustered or loose.

Doyle's algorithm basically consists in matching documents to existing clusters by computing a document-cluster score for each document relative to each cluster and admitting a document to those clusters for which a sufficiently high score is recorded. New centroids are then computed for each altered cluster. The process is then repeated; at each step of the iteration all the documents are correlated with all the clusters, and the clusters are updated until further updating does not alter any of the existing clusters.

It can be shown that Rocchio's algorithm requires order N^2 vector comparisons, where N is the number of documents, while Doyle's algorithm is of the order $N \cdot m$ where m is the number of clusters. A more efficient method due to Dattola [4] requires time proportional to $N \cdot p \cdot \log_p m$ where N is the number of items in the collection, m is the number of clusters desired and p is the number of clusters produced at each level of the algorithm. The method is an outgrowth of Doyle's attempt to obtain a fast algorithm for clustering large document collections. In each cycle of Dattola's algorithm, each document in the collection is scored against each existing cluster by a certain scoring function. New clusters are then computed while some documents remain loose. The cycle is then repeated with the new clusters. The algorithm is designed to control the number of clusters, size of clusters and amount of overlap. The number of clusters and amount of overlap are specified as input parameters, while the size of the clusters is controlled internally. One problem with Dattola's algorithm is that some way must be found to designate initial clusters.

An inexpensive one-pass clustering method has been proposed by Rieber and Marathe. [5] In this method the first document automatically becomes the centroid of the first cluster. Subsequent documents are correlated with existing clusters and depending on how the correlation with each cluster compares with the minimum correlation cut-off, the document is either admitted to one or more existing clusters or allowed to start a new cluster. If a document is admitted to an existing cluster, the cluster centroid is recomputed. The method allows for disjoint clusters where a document can only be included in one cluster, or overlapped clusters where a document is included in every cluster with which it has a high correlation. The single pass method of Rieber and Marathe compares favorably with more complex clustering methods in search cost, but there is no control on the cluster size or the number of clusters generated. This results in initial clusters being exceptionally large. Moreover, the process is likely to be order-dependent since the formation of the clusters depends on how the documents are encountered.

3. Strategy

To produce retrieval results for the user, two major costs are incurred: the cost of preparing the collection, and the cost of searching. Search cost can be reduced if an investment is made in clustering the collection. The aim of this study is to give a clustering algorithm which operates substantially more cheaply than those presently in use but for which the search costs are similar. In this way it is possible to compare clustering cost directly with search effectiveness. Alternatively, search parameters can be adjusted until search effectiveness is comparable, yielding a direct comparison of clustering and search costs. In either case, it may be possible to exhibit the extent to which the clusters are less optimal than those of other algorithms.

One approach to keeping search costs low is to use an algorithm which, within the constraint of a single pass over the document collection, will produce on the basis of document vector similarities a set of clusters of a given size distribution measured in terms of mean size, maximum size and overlap. This aim is achieved by the algorithm described in this study. The extent to which sets of clusters with similar size distributions have similar search costs is discussed in Section 6. C.

Specifically, the experiments are designed to allow a comparison with Dattola's algorithm. In his work [4], a mean cluster size is chosen. Clusters more than twice or less than half the mean size are not allowed. In the one-pass algorithm described in this study, the mean and maximum are easily controlled. The minimum is not controlled directly, since doing so requires blending small clusters in a second pass. It is questionable whether fixed limits on the cluster size are desirable. Certainly some sort of upper limit is needed to keep search costs well below that for a full search. The effect of a lower limit, given a mean and maximum, is less clear. In any event the method does not control the minimum whereas Dattola's does.

The composition of clusters from a single-pass method depends on the order in which the documents are processed. A document can not be placed in a cluster with a sequence number higher than that of the document. For example, if there are N documents in the collection and n clusters are produced, the first n-1 documents cannot belong to cluster n . In general, it will also be more difficult for a document N to join cluster 1 than for a document with an earlier sequence number.

The effect of order can be controlled partially and indirectly by controlling the rate at which clusters are allowed to form in the early part of the pass. Otherwise, order dependency is inherent in the single-pass method just as it is in other methods in which the number of iterations is limited. The degree of order dependency of the algorithm of this study is discussed in Section 6. B.

4. The Algorithm

The clustering algorithm accepts input vectors, each describing a document, and assigns each document either to one or more existing clusters or to a new cluster, depending on the correlation of the input vector with the cluster centroids. Each document vector is of the standard form, consisting of pairs of concept numbers and corresponding weights. The weight of each term in a centroid is obtained by summing the weights of that term for all documents in the centroid.

Vector correlations are computed as the multidimensional cosine between them, COS, as follows

$$\mathrm{COS} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{j=1}^{n} (u_j)^2 \sum_{j=1}^{n} (v_j)^2}} ,$$

where

COS = the cosine correlation between vectors u and v

n = number of terms in the collection dictionary.

COS ranges from 0 to 1. In order to control the cluster size, COS is modified, as discussed below. It is the modified correlation, COR, which is actually used in making clustering decisions.

One stage of the clustering process is defined as the comparison of one input document with all existing clusters. If the correlation, COR, of the document with the centroid of a cluster is greater than the cut-off value, CORCUT, the document is added to the cluster with which it has the best correlation. The document is also added to any clusters for which COR lies no more than an amount GAP below the highest correlation (maximum over all clusters) COR, thus producing overlap. If GAP = 0, no overlap occurs. If a document is not admitted to any cluster, it defines a new one. A centroid is recomputed whenever a document is added.

Number of clusters, cluster size and overlap are controlled dynamically. When, during the course of clustering, the number of clusters compared to the number of documents already processed becomes large, it is necessary to make it easier in general for documents to join clusters. However, to control cluster size it must be generally more difficult for a document to join a large cluster than a small one. To achieve these ends, CORCUT is varied to control the number of clusters while individual correlations, COS, are reduced by an amount related to cluster size in order to control cluster size near some maximum value.

A) Cluster Size

It is desired to define a function COR which will depend on the cosine correlation, COS, and also on the number of documents which the cluster would contain if the new document were admitted. COR should increase with COS and decrease as cluster size increases. If COS is very high, however, it would be unreasonable to exclude the document even from a large cluster. Therefore, when COS is 1, COR should equal 1 as well.

The following function, chosen for the algorithm, meets the above requirements:

$$\mathrm{COR} = \mathrm{COS}^{y}$$

where

$$y = \mathrm{NCEIL}/\ (\mathrm{NCEIL} - \min\ (\mathrm{NCEIL} - 1/B,\ M_i + M_c) + A)/A$$

NCEIL = cluster size ceiling requested by user

M_i = number of documents in input vector
(M_i = 1 unless clusters are being clustered)

M_c = number of documents in cluster

A,B = tuning parameters which the algorithm supplied and the user in general need not be concerned with.

Parameter A controls the rate at which the ratio COS/COR grows with cluster size when a cluster is small. Parameter B controls this ratio near and beyond the cluster size limit. If B is small, clusters can actually grow over the limit given by the user.

B) Number of Clusters

Following each stage, CORCUT is recomputed in order to control the number of clusters finally produced. The ratio of clusters produced to documents processed up to the moment is computed. If this ratio is larger than the value desired in the final result, CORCUT is adjusted downward from a base value FCOR, reducing the probability that a new cluster will be generated in the next stage. If the ratio is smaller than the desired value, CORCUT is raised. The base value FCOR moves toward CORCUT at a rate fixed by the algorithm. The user may supply a value for this rate.

New values of CORCUT and FCOR are computed as follows:

$$\mathrm{CORCUT}_i = (\mathrm{FCOR}_{i-1})^x$$

$$\mathrm{FCOR}_i = \mathrm{FCOR}_{i-1} * R + \mathrm{CORCUT}_i * (1 - R)$$

where

x = (NCL - NCLREQ * NINPUT / NE + D) / D

NCL = number of clusters following stage i-1

NCLREQ = number of clusters requested by user

NE = number of documents in source collection

NINPUT = number of documents input through stage i-1

R = parameter controlling rate at which FCOR follows CORCUT

D = tuning parameter set by algorithm.

If R=0, CORCUT varies widely and poorer clusters are produced.

C) Overlap

A document is added to a cluster only when two conditions are met:

1. COR > CORCUT
2. max(over all clusters) COR - COR < GAP

It can be seen that GAP controls overlap. Between stages GAP is adjusted as follows:

$$GAP_i = (FGAP_{i-1})^z$$

where

$$z = (M_1 + \dots + M_{NCL} + E)/(NINPUT*(OVLAP + 1) + E)$$

OVLAP = the value requested for

$$\left(\sum_{i=1}^{NCL_{NE}} M_i/NE \right) -1$$

FGAP = base value of GAP

$FGAP_i = FGAP_{i-1}*R + GAP_i*(1 - R)$

D) An Example

The following example illustrates how the clustering parameters are adjusted dynamically during the clustering process and how the individual correlations are computed. The values presented are taken from a run on the ADI collection. User selected parameters are given as follows:

NCEIL = 15

NCLREQ = 9

OVLAP = .122

Default options are used for the other parameters, namely:

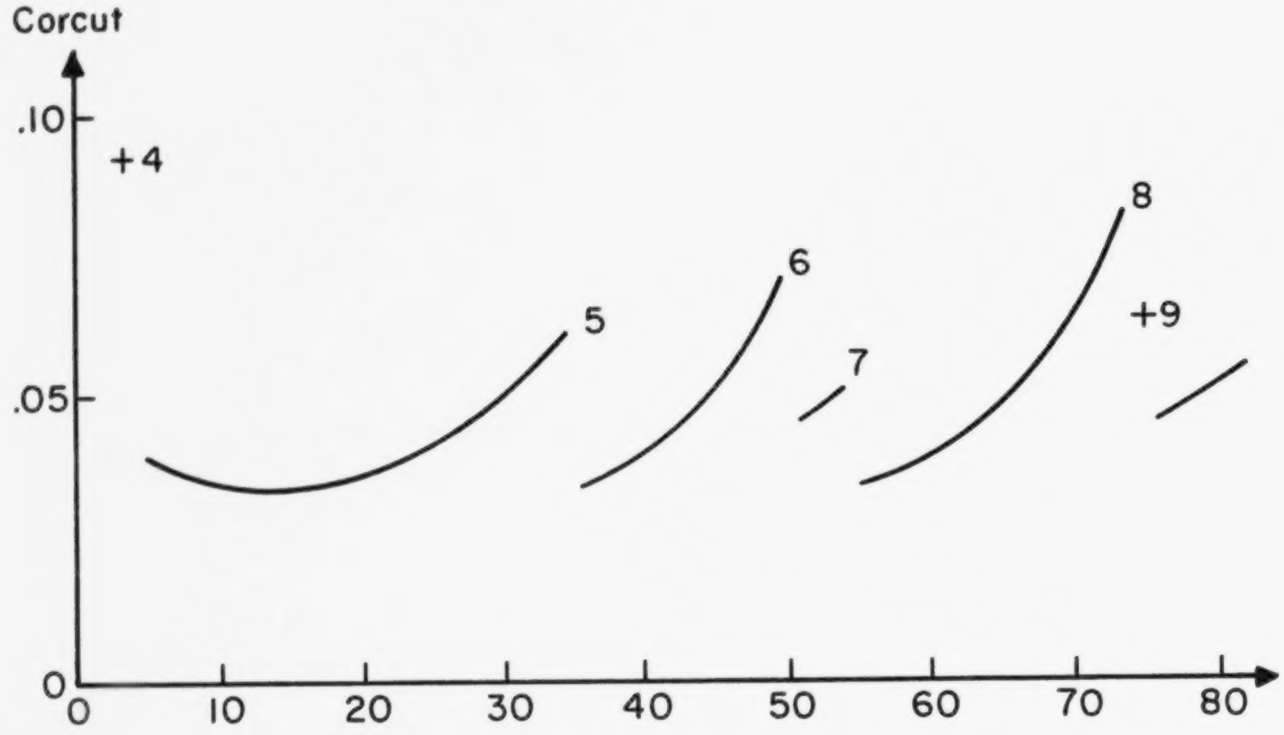
FCOR = .4
FGAP = .001
R = .9

A = 40
B = 2
D = 5
E = 1

Fig. 1 shows how CORCUT varies during clustering. Fig. 2 gives a similar presentation for GAP. Of course, CORCUT and GAP change in discrete steps after each document has been clustered. Points in the figures have been connected for ease of presentation.

Consider, for example, the input of the 65th document. Seven clusters exist at this point, so COS and COR are computed with respect to each cluster, giving the following results:

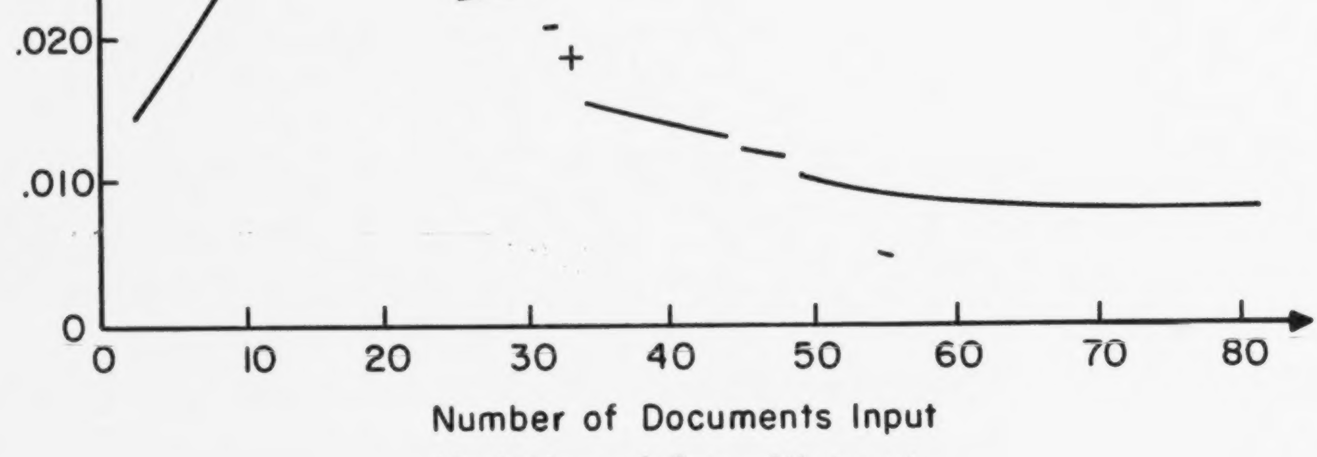
Cluster Number	Number of Documents in Cluster	COS	y	COR
1	14	.458	2.5	.142
2	10	.080	1.3	.037
3	14	.498	2.5	.175
4	12	.214	1.5	.010
5	12	.205	1.5	.009
6	1	.232	1.17	.181
7	1	.050	1.17	.030



Variation of Corcut with Input

Points at which new clusters form are marked with cluster numbers. First three points are off scale.

Fig. 1



Variation of Gap with Input

Each downward discontinuity in the plot occurs when a document is assigned to multiple clusters

Fig. 2

Taking the values of CORCUT = .045 and GAP = .0089 as adjusted following the 64th document, document 65 is admitted to clusters 3 and 6 as follows:

Cluster Number	Cor	Admit	
6	.181	*	
3	.175	*	
		<————————————	.181-Gap =.172
1	.142		
		<————————————	CORCUT = .046
2	.037		
7	.030		
4	.010		
5	.009		

5. Implementation

The algorithm is implemented in Fortran. The user specifies starting values as follows:

NCEIL = approximate maximum number of documents desired in any cluster

NCLREQ = number of clusters desired

OVLAP = fractional overlap desired

The algorithm sets the following parameters:

FCOR = .4
FGAP = .001
R = .9

A = 40
B = 2
D = 5
E = 1

The user may override these values if he desires. It is expected that extensive use of the algorithm would lead to better values than those already found, although the algorithm is not highly sensitive to them.

A) Storage Management

The algorithm is designed with a veiw toward application to large collections. Core storage and accesses to secondary storage are minimized in the following ways. Clusters are stored in sequential locations rather than in a 2-dimensional array. Only sufficient core storage for the clusters as a group need be alloted regardless of the variation in cluster size. A linked list could also be used. However, if secondary storage has to be used to store part of the cluster information, sequential storage is them preferable.

Sequential storage requires moving the cluster information in order to insert new concept-weight pairs. To minimize moves, two consecutive input vectors are kept in core. In the course of one stage of the algorithm, the input to the previous stage is added to the appropriate clusters and correlation of the current document is made simultaneously. The entire cluster collection is moved at most once for each stage.

6. Results

This study employs two document collections to evaluate the performance of the single-pass algorithm, the ADI collection of 82 documents and 35 queries and the Cranfield collection of 424 documents and 155 queries. Evaluation is made by comparing the results using the present algorithm with the results using Dattola's algorithm. Several clustering and search runs* were made using similar input parameters in both algorithms. Full search results on the ADI collection are also shown for comparison.

---

*In the recall-presicion curves presented, the modifications proposed by Dattola, [4] pp. 16-24, to compensate for variations in correlation percentage and uniform distribution of unrecovered relevant documents are incorporated.

The cost of clustering is first discussed in Section 6. A. Several clustering runs were made using the single-pass algorithm over different orderings of the ADI collection to show the effects of the order in which documents are processed. This is discussed in Section 6. B.

Clustering is also done at two levels as well as one level to illustrate savings in clustering time and to show the effect of multilevel clustering on cluster quality. The results of searching the ADI collection, including multilevel search, are discussed in Section 6. C. Finally, the algorithm is applied to the Cranfield 424 collection and the results are discussed in Section 6. D.

A) Clustering Costs

Cost comparisons between clustering methods can be made according to several criteria. The two for which results are presented in this study are:

1. Number of vector comparisons performed.
2. Computer resources used, mainly CPU and I/O time.

The first criterion allows comparison of algorithms to a large degree independently of the programming techniques employed and the system in which the programs are embedded. The second reflects the effects of the system and the programming techniques as well.

Consider the number of vector comparisons. It is convenient to assume that clusters are formed at uniform intervals during the processing of the collection, that is, cluster 1 forms with document 1 and in general cluster i forms with document $\frac{N}{m}(i-1) + 1$ and so forth, where there are N documents and m clusters produced. Under this assumption, the number of comparisons for clustering on a single level is given as follows:

$$C_1 = \frac{N - 1}{m} \sum_{i=1}^{m} i = \frac{(N - 1)(m + 1)}{2}$$

where

C_1 = number of comparisons on level 1

N = number of documents in the collection

m = number of clusters produced

If clustering is done on x levels and p clusters are formed at each level from each cluster on the next higher level, the number of comparisons made at level i is

$$C_i = \frac{(N - 1)(p + 1)}{2} \quad .$$

Consequently, for multilevel clustering over x levels, the total number of comparisons C is

$$C = \sum_{i=1}^{x} C_i = \frac{(N - 1)(p + 1)}{2} x$$

and since $m = p^x$,

$$C = \frac{(N - 1)(p + 1)}{2} \log_p m \quad .$$

Dattola gives a similar derivation in complete detail [4], giving the total number of comparisons D over x levels as defined above:

$$D = kNp \log_p m$$

where

k = the average number of times a set of documents is compared to a set of cluster centroids.

A typical value of k is 16 for a collection the size of the Cranfield 424.

Based on vector comparisons, then, one could expect an increase in speed over Dattola's algorithm of 2k for large p and $\frac{3}{2}$ k for p = 3 (the optimum value proposed by Dattola [4]). A somewhat lower ratio is shown in the results of this study. Table 1 shows comparative results. For the ADI collection the ratio of number of comparisons is 15:1, implying a k for Dattola's algorithm of 7.5. For the Cranfield 424 the values are 8.2:1 and 5.25, respectively. The difference from the value k = 16, which is expected, is largely explained by the following factors:

1. In the runs using the present algorithm, a burst of clusters was forced to form at the outset. Thus the uniform formation assumption is not met and more comparisons are made with the present algorithm than predicted. In the limiting case where the first m documents form the m clusters, C is bounded as follows:

   $$C < (N - 1)p \log_p m \quad .$$

2. During an iteration of Dattola's algorithm the number of trial centroids is frequently less than the chosen value of p .

ADI Collection (82 documents)	Present Study	Dattola's Algorithm
Number of clustering runs over which the following results are averaged	11	1
Vector comparisons	445	6614
CPU seconds (360/65)	5.8	44.0
I/0 seconds (360/65)	13.3	38.0

Cranfield 424 Collection* (424 documents)	Present Study	Dattola's Algorithm
Number of clustering runs over which the following results are averaged	1	1
Vector comparisons	5579	45840
CPU seconds (360/65)	214	439
I/0 seconds (360/65)	126	653

Clustering Cost Comparisons Between the Present Study and Dattola's Algorithm

Table 1

---

*Results shown for Dattola are for two-level clustering

3. Iterations of Dattola's algorithm sometimes converge before the iteration limit is reached.
4. Two levels of clustering were performed on the Cranfield 424 under Dattola's algorithm against a single level for the algorithm of this study.

The comparison of CPU times in Table 1 is somewhat less favorable to the algorithm of this study, particularly on the Cranfield 424 collection. The major factor is scoring, or vector comparison. Scoring may be done over an array of weights if concept numbers are restricted to a prescribed range. In the case of Dattola's implementation, numbers do not exceed 10,000 so comparison may be done over a 10,000-element array in which the concept number is given by position rather than in a list where the concept numbers appear explicitly. The present algorithm uses such a list, ordered by concept number, and runs more slowly as a consequence.

Perhaps the most important point to be made in this discussion is that the algorithms of Dattola and of the present study are of the same order. The constant multiplier k, however, is of the order 16 in the case of Dattola and 1 in the case of the present study.

B) Effect of Document Ordering

The effect of ordering is studied by comparing the search results on the original ADI collection and three permutations of it. The three permutations are constructed by reordering the collection according to tables of random numbers between 0 and 99.

Clusters are generated using NCEIL = 22, NCLREQ = 9 and OVLAP = .122 (default options are used for the remaining parameters). A minimum of 10 documents is searched for each query. Plots of precision vs. recall are shown in Figures 3 and 4. As expected, there is significant variation in the performance of the algorithm for different orderings of the documents. It is interesting to note that the original ordering of the ADI collection gives the worst results. The third permutation gives the best results which actually exceed Dattola's results except in the high recall region. It can be seen that the relative improvement of the results by reordering is better in the document-level average plots than in the recall-level average plots.

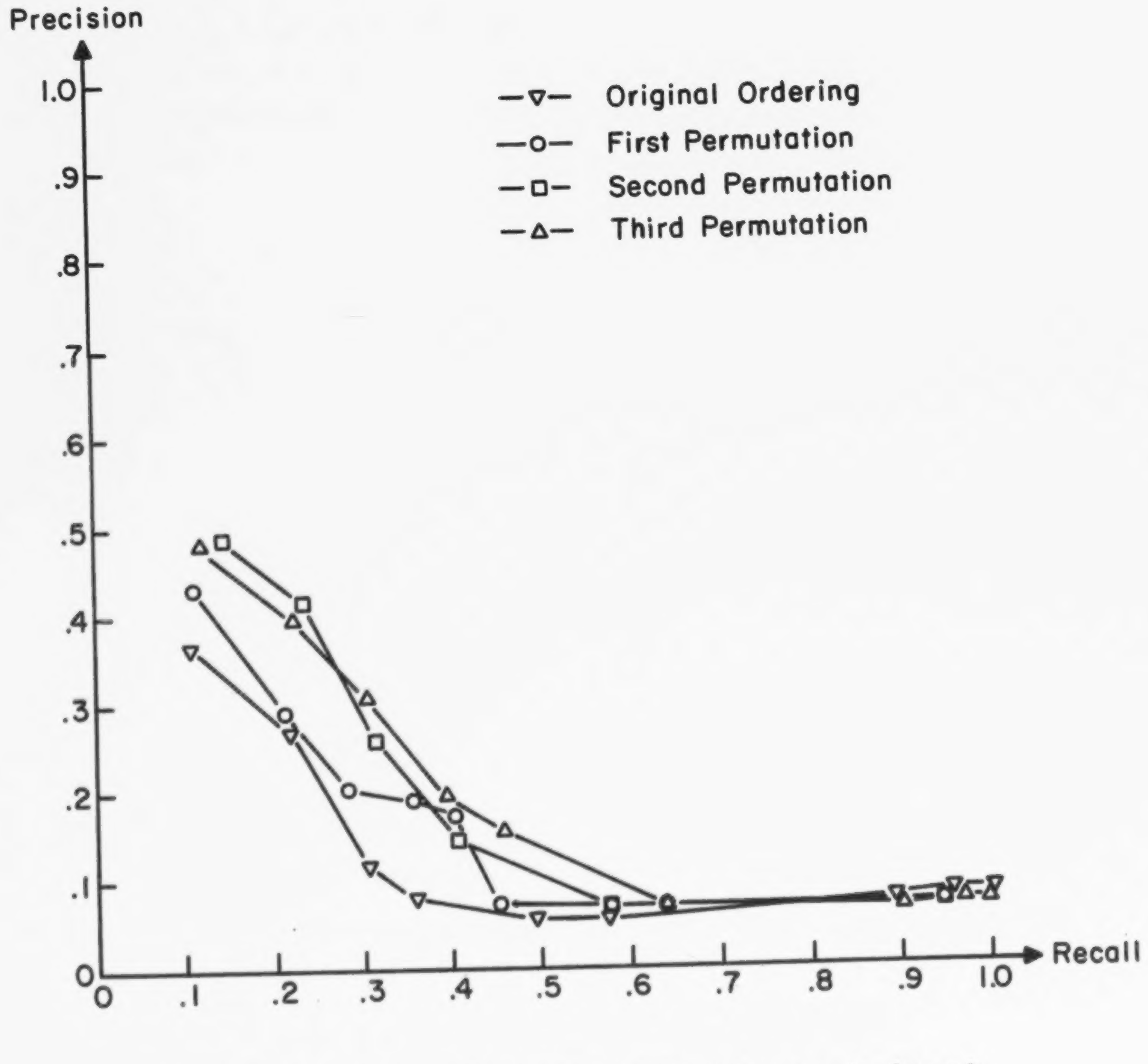
Figures 5 and 6 show the spread between the curves representing the original ordering and the third permutation of the documents of the ADI collection. The results can be compared with those of a full search and clustered search using Dattola's algorithm.

Multilevel clustering is discussed in Section 6. C. Since multilevel clustering improves search results, it is possible that document reordering may be used to obtain further improvement in this case, although it would be difficult to determine a suitable ordering in advance.

C) Search Results on Clustered ADI Collection

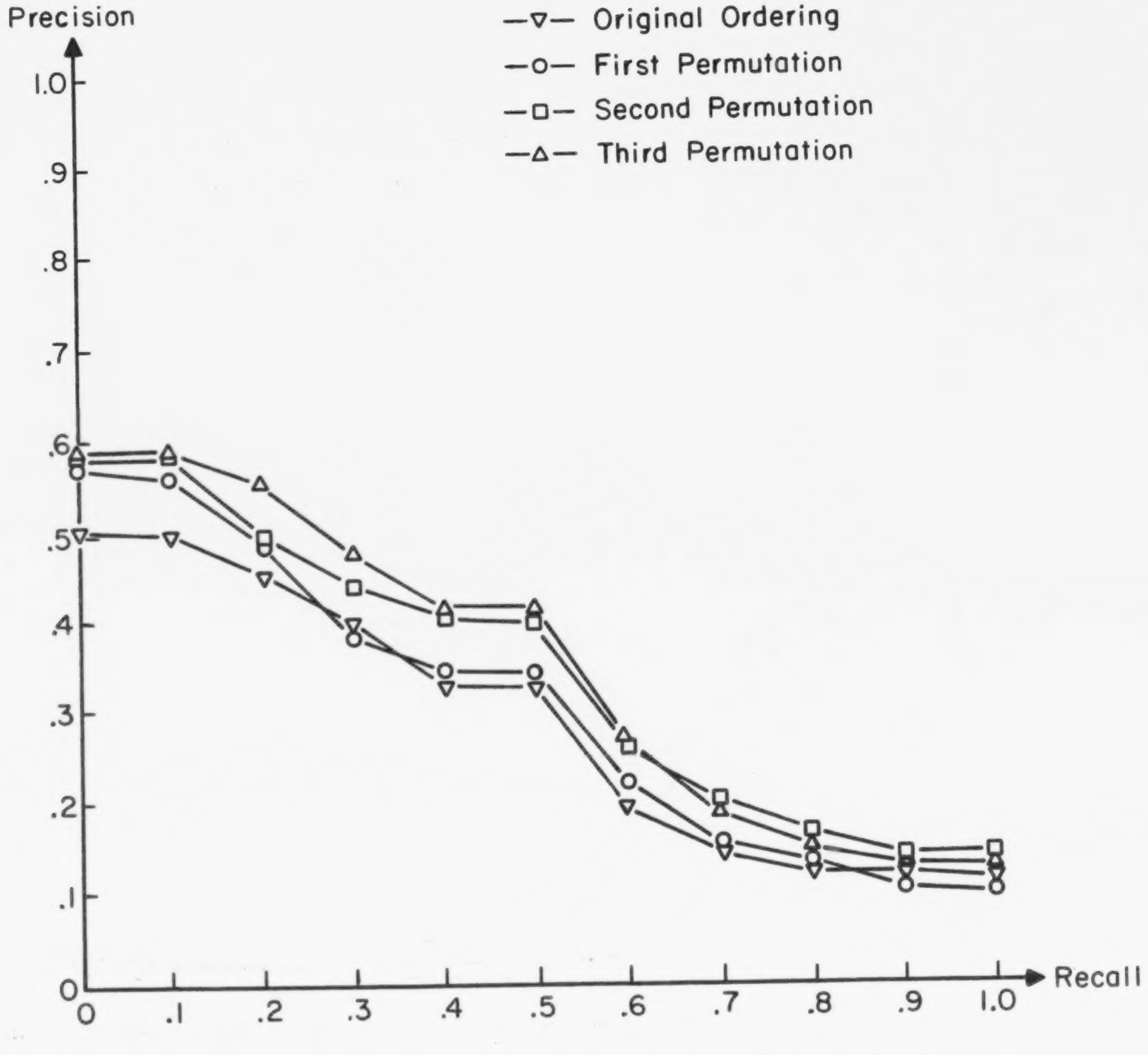
Recall and precision plots of searches on the clustered ADI collection are shown in Figures 5 and 6. As discussed in Section 6. B, it may be seen that there is a substantial variation in the quality of the clusters over different orderings of the collection, as measured by search results. In comparison with both the full search and Dattola's algorithm, the present algorithm shows a tendency to perform best in the low-recall region. This effect may be observed in all results of this study and, consequently, it is a distinguishing characteristic of the single pass algorithm.

It should be observed that search costs for the results shown using



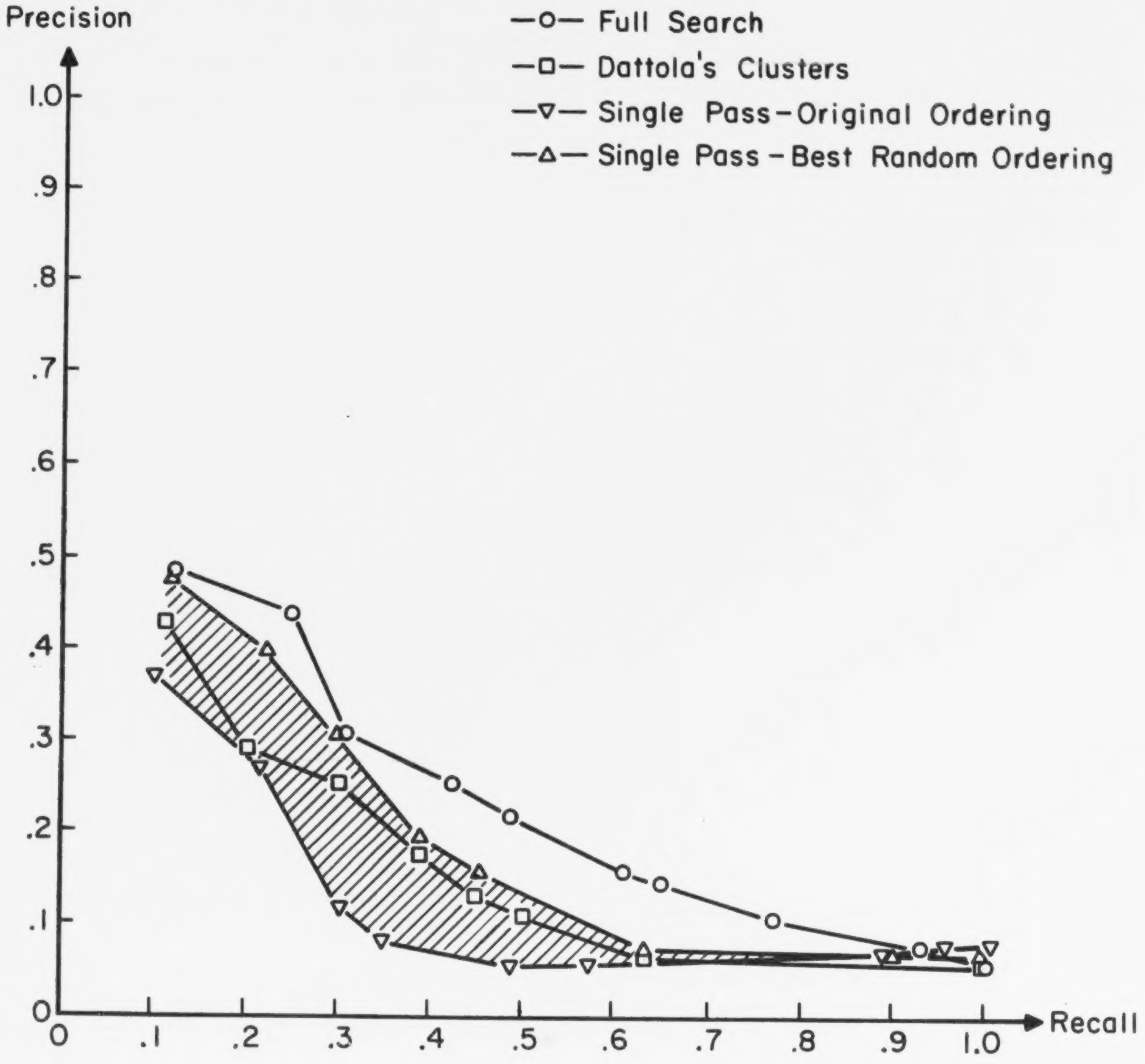
Effect of Ordering on Document Level Averages in Search of ADI Collection Clustered with Single Pass Algorithm

Fig. 3



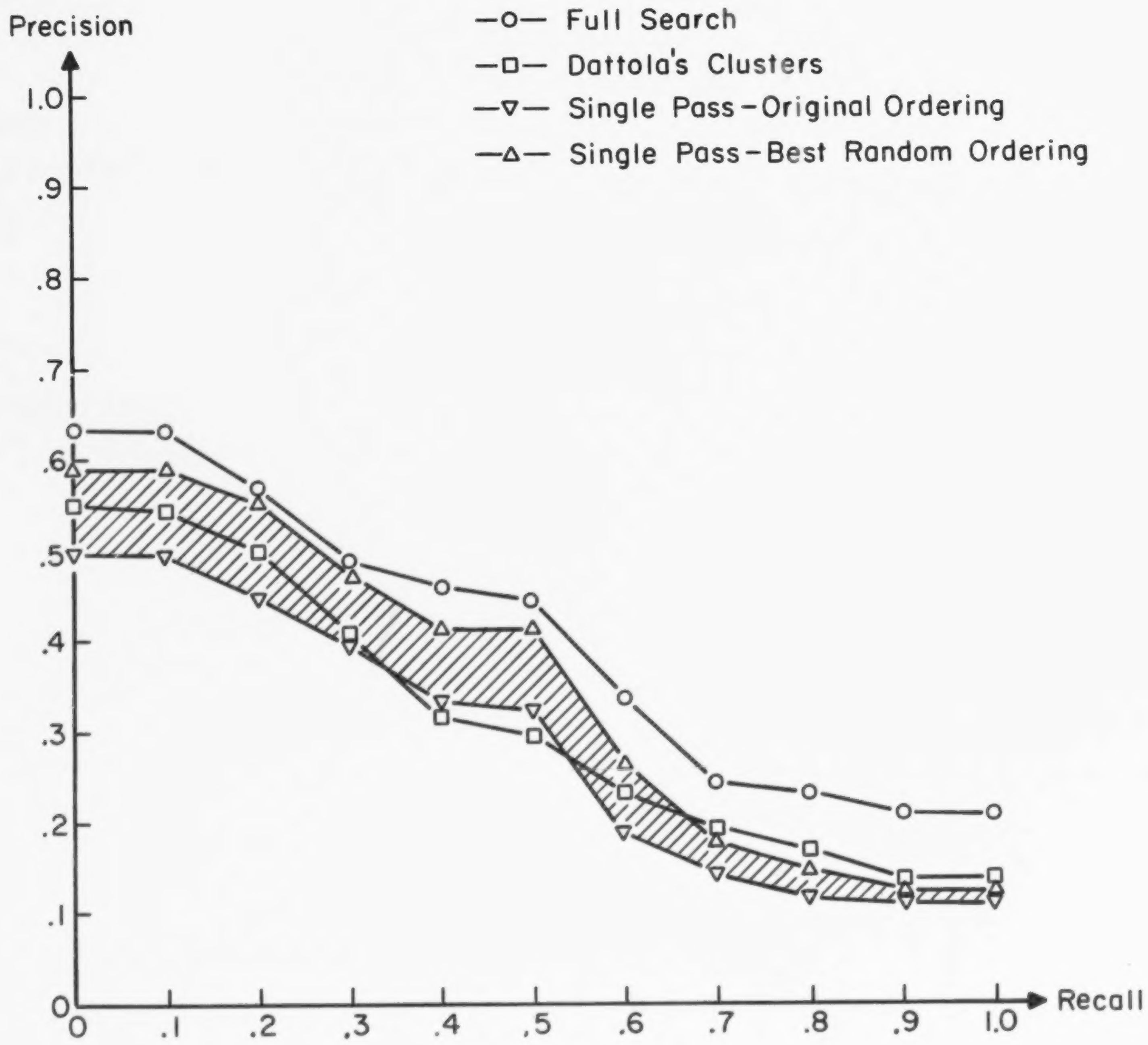
Effect of Ordering on Recall Level Averages in Search of ADI Collection Clustered with Single Pass Algorithm

Fig. 4



Document Level Averages for Full Search, Dattola and Single-Pass Clustered Search on ADI Collection

Fig. 5



Recall Level Averages for Full Search, Dattola and Single-Pass Clustered Search on ADI Collection

Fig. 6

the present algorithm and the results using Dattola's clusters are roughly similar. For example, 990 vector comparisons are made in searching the clusters produced from the ADI collection as originally ordered compared with 966 vector comparisons in searching Dattola's clusters.

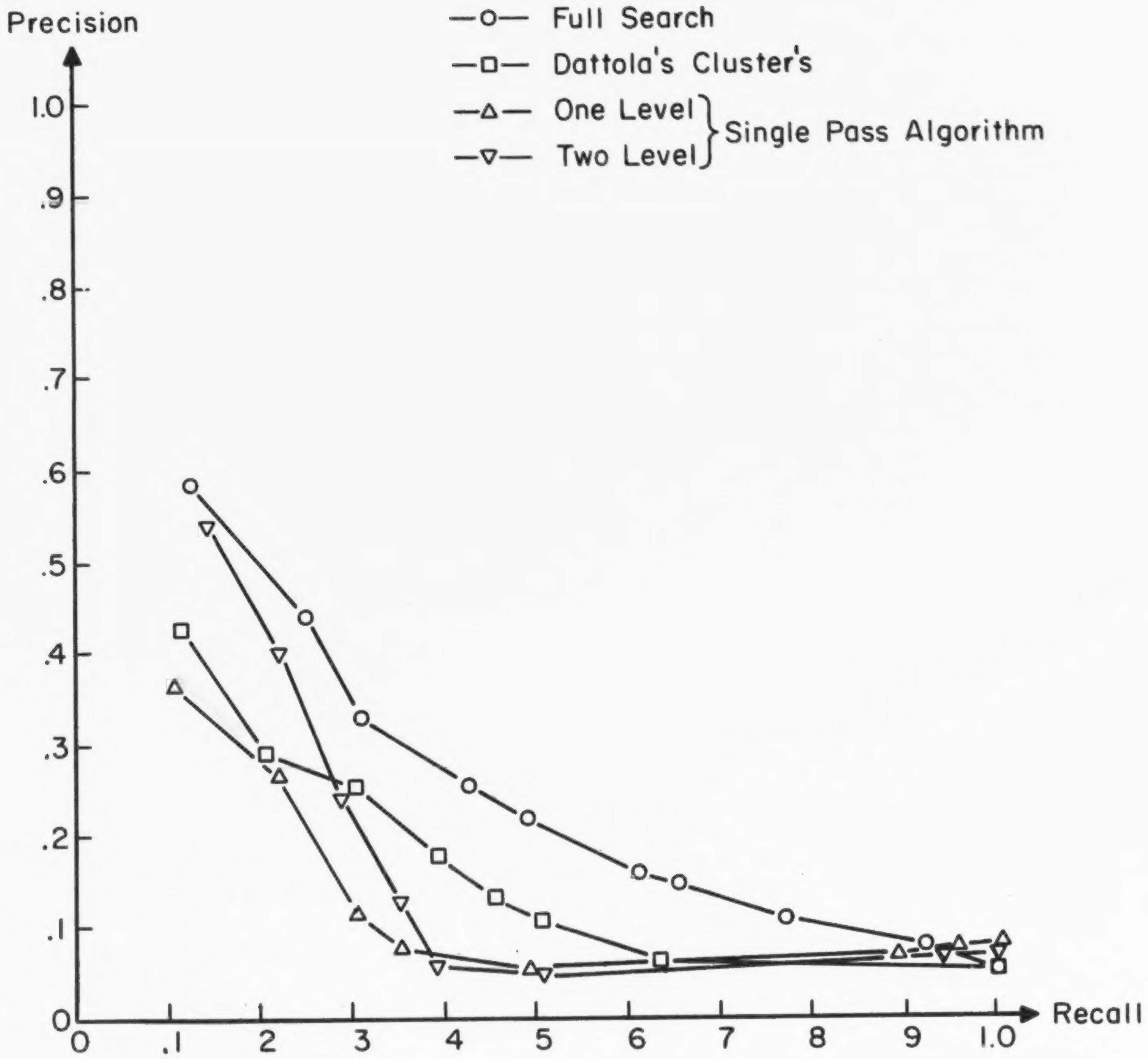
In Section 6. A an expression for the cost of clustering is given. The relationship is such that clustering cost is reduced both for the algorithm of this study and for Dattola's algorithm if multilevel clustering is done.* It is not known how the search results on clusters formed at a single level and over multiple levels compare in the case of Dattola's algorithm. However, in the case of the present algorithm, multilevel clustering is not only less expensive to perform but it can produce markedly better clusters.

Figures 7 and 8 show the improvement in recall and precision achieved when the same ordering of the ADI collection is clustered over one level and two levels. Clusters are formed at the first level and then sons of each are formed at the second level. The ordering chosen was the one among the four tested for which recall and precision are poorest for the single level clusters. It is suspected that improvement would also be shown for the other orderings.

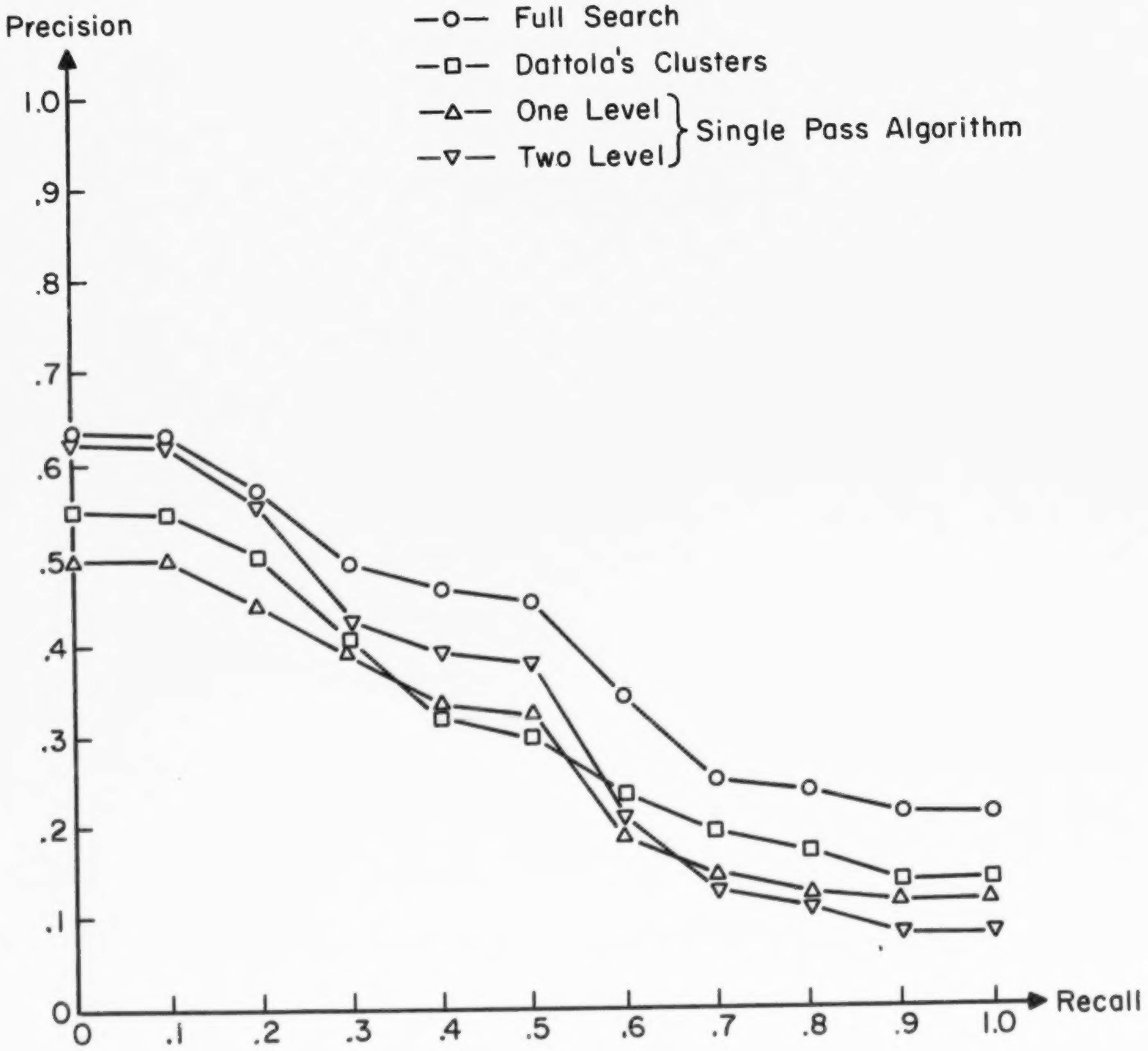
It can be noted in Figures 7 and 8 that the two-level clustered search is markedly better than the one-level search, particularly in the low recall region. Moreover, the two-level search performs better than the search on Dattola's clusters in the low recall region and approaches the full-search curve in this region.

---

*It is thought proper to apply the description "single pass algorithm" to multilevel clustering where (a) each level is clustered in a single pass and (b) the multilevel algorithm performs fewer comparisons than a single pass would perform as a single level.



Document Level Averages for Full Search, Dattola, One-Level and Two-Level Clustered Search of ADI Collection
Fig.7



Recall Level Averages for Full Search Dattola, One Level and Two Level Clustered Search of ADI Collection
Fig. 8

In general, the probability that a document late in the input enters an early cluster is related to the number of clusters formed in the pass over a level of the cluster tree. The greater the number of clusters, the more the membership of early clusters is confined to documents early in the collection. This can be seen by observing that, except for the case where a collection is partitioned into m sequential clusters, a cluster on the average allows documents to be admitted over a sequence of input documents larger than the cluster size, that is, for a single level of clustering,

$$n = a \frac{N}{m}$$

where

n = number of documents for which a substantial probability of admission to a certain cluster exists.

a = a constant $(a > 1)$.

If clustering is done over two levels,

$$n_j^{(2)} = a \frac{n^{(1)}}{p}$$

where $n_j^{(2)}$ is the jth son of $n^{(1)}$ (superscripts refer to the level of the tree). Thus, at the final level x there are m clusters and

$$n^{(x)} = a^x \frac{N}{p^x} = a^x \frac{N}{m}$$

$$n_j^{(x)} > n_j^{(1)} \quad \text{for} \quad x > 1 \quad \text{and} \quad a > 1$$

The above suggests that producing as few as two clusters from each father in the cluster tree would allow the best associations to be made. It is also suspected that better results may be obtained if the order in which the documents are compared at each level is reversed. The rationale here is that the last documents admitted to the cluster have in general the highest correlation with the centroid. It is hypothesized that in a pass in reverse order, these documents will tend to form a single cluster and allow the earlier ones to fall in separate groupings.

The search results reported above suggest that sets of clusters with similar size statistics have similar search costs when the same search parameters are used. Comparative figures are shown in Table 2.

Overlap figures are also given in Table 2. It may be seen that overlap varies substantially between the sets of clusters compared. Certainly, increased overlap increases search costs since it increases average cluster size. Whether increased overlap affects recall and precision curves to any great extent is less clear. It may be argued that the clustering algorithm operates without knowledge of the query set subsequently used to search the collection and, consequently, assignment of documents to multiple clusters is independent of relevance judgments. Unpublished results of Dattola in which overlap was varied widely when clustering both the ADI and Cranfield 200 collections without apparent effect on recall and precision support the hypothesis of independence.

The results of this study as well suggest that overlap is uncorrelated with recall and precision. Overlap is not held constant in the results presented because of the difficulty in matching the overlap measures of Dattola's algorithm and of the algorithm of this study. As may be seen

	ADI Collection (as ordered)			Cranfield 424	
	Single Pass		Dattola	Single Pass	Dattola 2-level
	1-level	2-level	1-level		
Fractional overlap	.17	.33	.01	.14	.23
Average cluster size (No. of documents)	10.7	12.1	9.2	24.2	23.7
Average cluster size/ collection size	.13	.148	.112	.057	.056
No. of clusters	9	9	9	20	22
No. of clusters/ collection size	.11	.11	.11	.047	.052
No. of vector comparisons (Search Cost)	990	1105	966	13,103	12,200
Fractional search cost	.35	.39	.34	.20	.186
No. of documents sought in search	10	10	10	43	43

Summary of Cluster Statistics
and Search Costs

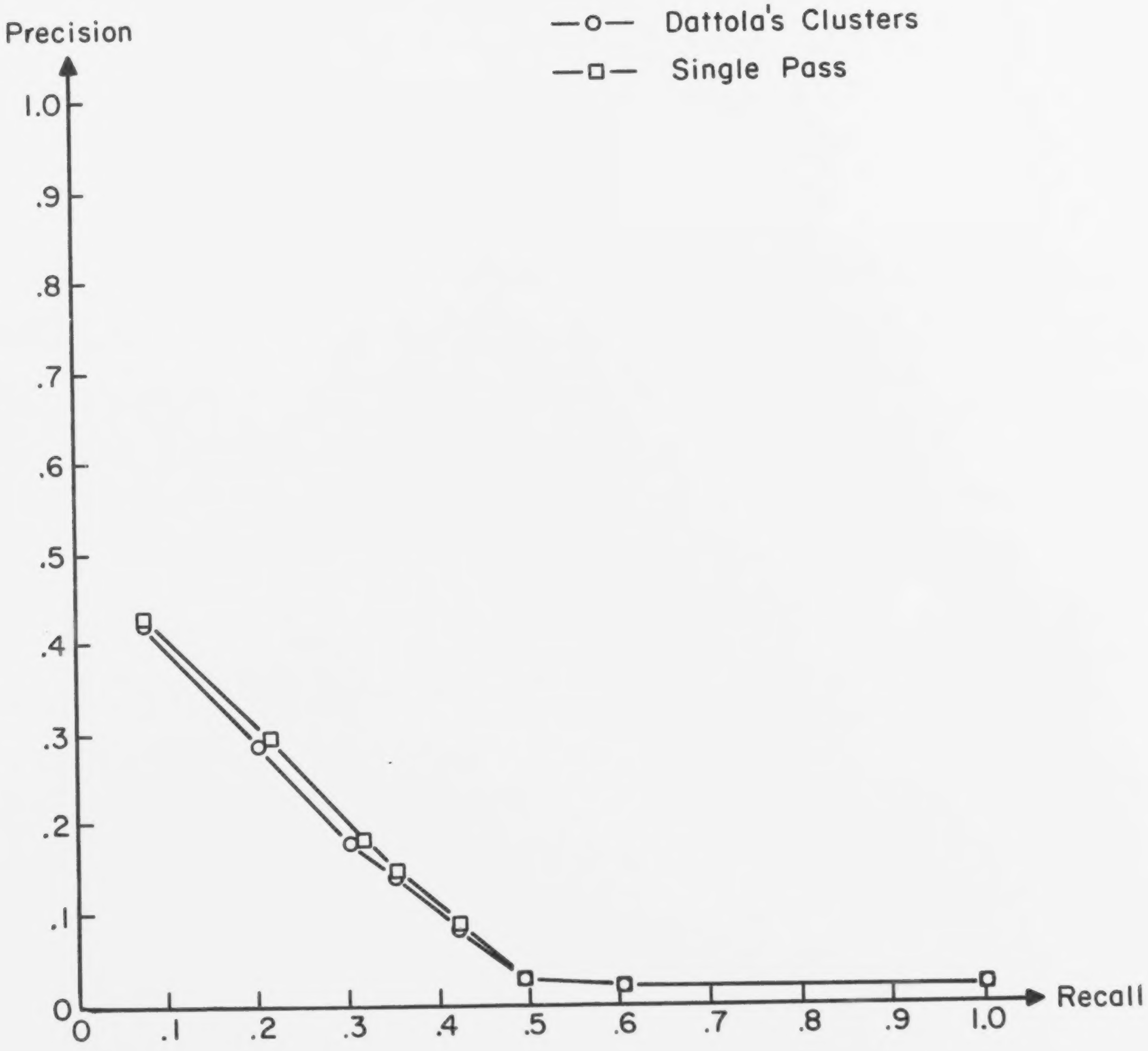
Table 2

in Table 2, higher overlap values occurred with the better results on the ADI collection, but the reverse is true for the two runs made on the Cranfield 424 collection.

D) Search Results of Clustered Cranfield Collection

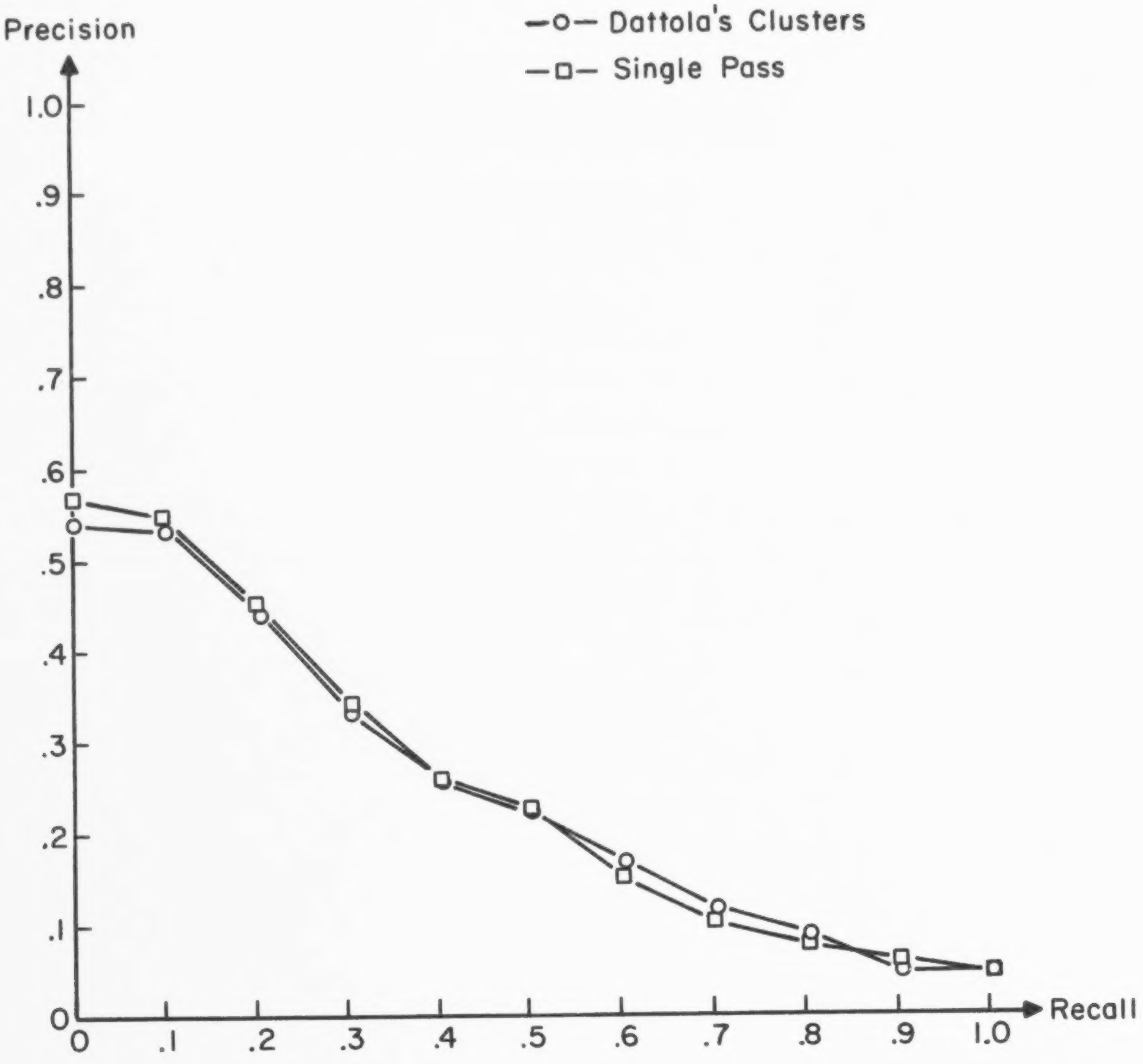
The Cranfield 424 collection is known to have sequential groupings of some of the documents relevant to certain queries. Consequently, a random ordering of the collection is used as input to the single level clustering run performed with the algorithm of this study. Figures 9 and 10 show the recall and precision curves from a search on this set of clusters compared with a search on a set of clusters produced with Dattola's algorithm. In the case of Dattola's algorithm [4] clustering is done over two levels using clustering parameters which were found to be optimal on the ADI and Cranfield 200 collections.

As may be seen from Figures 9 and 10 the algorithm of this study produced a slightly better recall and precision curve than Dattola's. Additional runs on several permutations of the collection are needed to establish whether a significant difference is shown consistently. Search costs on the Cranfield 424 collection using the single pass algorithm are comparable to search costs using Dattola's clusters. As shown in Table 2, the present algorithm requires 13,103 vector comparisons, compared with 12,200 for Dattola's case. A more useful measure, which allows comparison of collections of different size, is the fractional search cost, defined as the number of vector comparisons per document per query. Fractional search costs on the Cranfield collections for the single pass method and for Dattola's are roughly similar, as seen in Table 2.



Document Level Averages for Dattola and Single Pass Clustered Search of Cranfield 424 Collection

Fig. 9



Recall Level Averages for Dattola and Single Pass Clustered Search of Cranfield 424 Collection

Fig. 10

7. Conclusions

The single pass algorithm of this study is substantially less costly to execute than other clustering algorithms, Dattola's in particular. As may be expected, however, the quality of the cluster set depends to a large degree on the order in which the documents are encountered by the algorithm. On the ADI collection, some orderings produce search results, measured by recall and precision curves, better than Dattola's clusters in the low recall range. Other orderings are worse at virtually all points on the curve.

A single clustering run is reported for a larger collection, the Cranfield 424. It indicates that cluster quality is not degraded when the algorithm is applied to a collection about five times as large as the ADI.

Multilevel clustering using the basic single pass approach of this study is shown both to be cheaper and to produce substantially better clusters as measured by search results. There is a strong suggestion that further work could establish the multilevel single pass algorithm to be as good or better than Dattola's algorithm for most orderings of the collection.

The basic limitations of the single pass method appear to be overcome best when multilevel clusters form a binary tree. It is possible that the contents of a collection would dictate nodes in the tree of higher degree. The trade offs involved in such cases should be investigated.

The multilevel clustering of this study is confined to presenting only the lowest level of the tree to the search algorithm. However, the entire tree could easily be made available to the search algorithm. If so, two possible ways of construction present themselves. The first is to fetch each document description from the collection only once. Each document would be passed down the tree and all decisions relative to it would be made in sequence, level by level. In effect, many levels of single pass clustering would be carried on simultaneously, each cluster at each level being treated as the collection from which its sons are formed. In the case of a binary tree, for example, the first document in the collection would be passed down the entire leftmost branch of the tree to the predetermined lowest level. It would define, at first, the leftmost cluster at each level. The next document would then be passed down the tree. As long as it was admitted to existing clusters it would add to their definition. At the point (if any) where it was selected to define its own cluster, it would form a right branch and then a sequence of left branches down to the lowest level. Subsequent documents would be processed similarly.

The second method of construction is to form each level completely before the next level is begun. Document descriptions would have to be fetched from the collection once for each level plus additional occurrences caused by overlap. However, cluster quality might well be better since the direction of passing over the documents can be reversed at each level.

Even when just two sons are formed from each father in the tree, there still exists the possibility that natural clusters will be split into fragments. By the nature of the process, once two documents are separated, they cannot be associated again. To a certain degree, searching over multiple clusters allows these documents to be found. It would be best, however, to have them properly associated in the tree. It is proposed, therefore, that the leaves of a completed tree be compared one to another. Those with particularly high correlations would be coalesced into a single cluster taken to be the son of both fathers. Such a coalescing process would deform the tree only at the lowest level and could be expected to reassociate sets of documents which were of roughly equal size and large enough to be a majority of the members of the clusters involved. Any further investigation will necessarily include experiments designed to strengthen, if possible, the results already found and to consider further the selection of optimum clustering parameters.

References

[1] D. Williamson, R. Williamson and M. Lesk, The Cornell Implementation of the SMART System, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section I, Department of Computer Science, Cornell University, September 1969.

[2] J. J. Rocchio, Jr., Document Retrieval Systems: Optimization and Evaluation, Harvard Doctoral Thesis, Information Storage and Retrieval, Report ISR-10 to the National Science Foundation, Harvard Computation Laboratory, Cambridge, March 1966.

[3] L. B. Doyle, Breaking the Cost Barrier in Automatic Classification, SDC paper SP-2516, July 1966.

[4] Dattola, R. T., Experiments With a Fast Algorithm for Automatic Classification, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section XIII, Department of Computer Science, Cornell University, September 1969.

[5] V. P. Marathe and S. L. Rieber, A One-Pass Clustering Algorithm, Information Storage and Retrieval, Report ISR-16 to the National Science Foundation, Section XIV, Department of Computer Science, Cornell University, September 1969.

XIII. A Systematic Study of Query-Clustering Techniques: A Progress Report

S. Worona

Abstract

An experiment using various techniques of query clustering on the Cranfield 424 document collection is described and some preliminary results are given. Several methods of evaluating the performance of clustered searches in the context of query-clustering are discussed. Finally, some observations are made concerning use of the SMART system as implemented at Cornell University.

1. Introduction

The idea of query-clustering as an aid to information retrieval systems is first defined and examined by V. R. Lesser [1]. In that report, a two-level clustering algorithm is described in which members of a document collection are assigned to clusters according to their relationship with previously-formed clusters of queries.

It is argued that there are three advantages to performing the clustering in this manner; first, the accuracy of a given search procedure may be increased by comparing queries to sets of related queries already processed by the retrieval system, instead of sets of related documents. Second, it is likely that such a system will perform better as time passes and more queries are available for clustering. Finally, because the cost of most clustering algorithms increases with the size of the collection being clustered, it is more economical to use a query collection than the associated — and much larger — document collection. A more thorough general discussion of the first two of these points can be found in [2], as well as in the original paper by Lesser. Applications of these ideas to various methods of query clustering are discussed below. (For additional information on clustering algorithms, see [3,4,5,6,7,8]. Background material and further references may be found in [9].)

The general process of query clustering may be divided into three parts, or "phases":

1) generation of query clusters;
2) generation of document clusters from the query clusters of phase 1; and
3) definition and assignment of centroids for the phase 2 document clusters.

Each of these three phases may be accomplished in several different ways. A combination of three such methods — that is, one for each phase — is termed an "implementation" of query clustering, or a particular query-clustering technique.

Many procedures exist for performing phase 1, that is, in the initial clustering job. The variables in using these algorithms include the number of clusters desired, the amount of overlap permitted, and the number of queries to be clustered. The last parameter is particularly important to a query-clustering technique, because it is hoped that search results improve with an increase in the number of queries clustered.

Phase 2 may be implemented in any of the following three ways:

1) Relevancy decisions
2) Correlation with query centroids
3) Correlations with clustered queries.

In the first case, documents which are relevant to one or more queries in any one query cluster are grouped. For case 2, all documents are correlated against each query centroid, and those documents correlating highly with any such centroid are put into one cluster. In the third case, the documents are correlated with all queries used in clustering, and a document cluster is formed from those correlating highly with one or more queries in any given query cluster. In all three methods, one document cluster is formed for each query cluster generated in phase 1. The query cluster from which such a document cluster is formed (whether by relevancy decisions, centroid correlations, or query correlations) is called its *underlying query cluster*.

There is much disagreement about the manner in which a centroid is chosen to represent the documents in a cluster. Concepts may be logical or weighted, with very high or very low weights arbitrarily either retained or dropped by one of several methods [10]. In query clustering, however, the choice of the type of centroids used (phase 3) is much more basic — the centroids for each phase 2 cluster may be either the *document centroid* formed from the documents of the cluster, or the *query centroid* of the underlying query-cluster.

There are obviously a large number of query-clustering techniques which may be formed from different combinations of the above variations of the three phases. At this time, the only available studies of query clustering are focused on varying phase 1 methods, while using relevancy decisions for phase 2 and query centroids in phase 3. This paper deals with an experiment which considers all six combinations of second- and third-phase variations, while also using different numbers of queries in phase 1. This produces eighteen different query-clustering techniques, which are then compared to a standard full search, and also to a set of "normally-generated" clusters (formed using Dattola's Algorithm). Because of the large volume of data generated by the experiment, a thorough analysis of the results is not yet available. The present report will be followed by such an analysis, using the parameters developed in [2].

2. The Experiment

A) Splitting the Collection

In all experiments with query clustering, it is first necessary to divide the query collection used into two disjoint subcollections: a cluster-set and a test-set. (The collection used in this case is the 155-query set associated with the Cranfield 424 documents.) In general, the cluster-set provides the queries for clustering; these clusters are then used to generate phase 2 document clusters. When a tree (that is, a hierarchy of documents and centroids) has thus been formed, the test-set queries are used to determine the performance of the particular method used. This simulates the action of an actual information-retrieval system, and makes clear the requirement that the two query-sets be disjoint. (Since one of the hypotheses being tested states that new queries entering a system benefit by the presence of similar queries already processed, it would be unrealistic to test methods of query clustering which allow the "new" queries to be present in the system already.)

Because of the nature of the query collection used, another consideration in this splitting is the authorship of the queries. In a real system, it is unlikely that a given author would submit two queries with similar sets of relevant documents. (If this were the case, relevance-feedback from the results of the previous query should best be used in handling the later query.) The Cranfield queries, however, contain many instances of authors' submitting several queries with similar sets of relevant documents. It is not unreasonable, then, that in splitting such a test collection into two sets of queries, it should be required that author-sets not be broken. That is, if any query of a given author appears in one of the sets, then all queries of that author are put into that set.

With these considerations in mind, the 155 queries are split into two author-consistent sets of 30 (test-set) and 125 (cluster-set) queries each. This is done by generating random numbers between 1 and 155, and including in the test-set those queries whose numbers are drawn at random, as well as all other queries by the same author. Random numbers corresponding to queries already selected are passed by in subsequent drawings. Appendix A, Table A1 gives the results of this splitting, including author number for all queries selected.

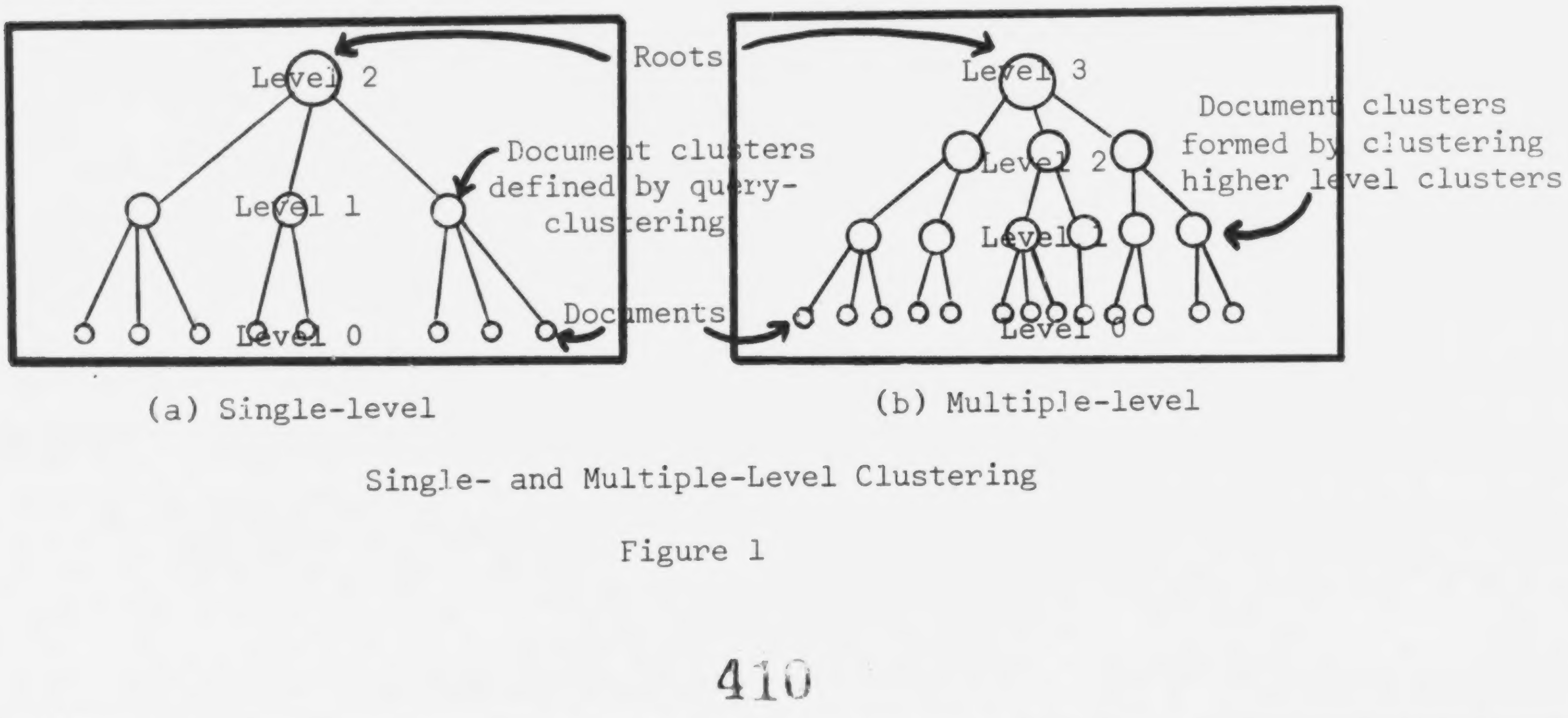
After the generation of the test-set, the cluster-set is formed from the remaining queries. In order to allow the experiment to test the effect of enlarging the base of clustered queries, the 125-item cluster-set is subdivided randomly into sets of 75 and 100 queries, such that the first is a subset of the second. Three cluster-sets — called CS1, CS2, and CS3 — are thus formed — with 75, 100, and 125 queries, respectively — where the increase in size from one to the next is caused only by the inclusion of additional queries. This, too, mirrors the action of a real system, where an increase in queries processed is caused by addition, rather than by reformulation. Table A2 in Appendix A lists the queries included in these three cluster-sets.

B) Phase 1: Clustering the Queries

The three sets of cluster—queries formed by splitting the Cranfield 424 collection queries are clustered using Dattola's algorithm (see [7]). Essential to this algorithm is a specification of the number of clusters desired and the amount of overlap permitted. The results of the experiment in [2] indicate that overlap in such a set of query clusters in greatly magnified when the related document collection is formed — particularly, when relevance decisions are used in phase 2. It is apparent, moreover, that overlap will also be increased by most other implementations of phase 2. Since the overlap obtained in [2] was far too great, it was decided that the query clusters formed here should have no overlap.

Not so easy to answer is the question of how many clusters should be formed. This is a problem in any one-level query-clustering technique. First, as many query-clusters must be formed as the number of desired document clusters. Furthermore, the number of queries to be clustered is generally far less than the number of documents. Thus, the number of clusters cannot be optimal for both the queries and the documents. According to [8], the best number of equal-sized clusters which can be formed from n items is of the order of $\sqrt{\underline{n}}$. Thus, for the 424 documents of the Cranfield collection, approximately 21 clusters should be made, while the three cluster-sets of queries require, roughly, 9, 10, and 12 clusters, respectively. One solution to the problem is to abandon the single-level hierarchy in favor of a multiple-level tree, where the clusters on level 1 are formed by breaking up the query-clustering-generated clusters on level 2. (see Figure 1.) This method is currently under investigation by Magliozzo and Bodenstein [11].

Because this experiment is not designed to consider such variations in query clustering, a solution other than that mentioned above is desirable. A compromise, therefore, is made between the different optimal numbers of clusters required by the four collections. Dattola's algorithm is asked to provide 15 clusters for each set of queries and documents to be clustered. The 15 clusters of the 424 documents thus generated are later used as a "standard clustering" against which the query-clusterings are compared. (The actual generation of so precise a number of clusters is not a trivial matter, as is discussed in Appendix C.) The results of using Dattola's algorithm to cluster these collections are given in Appendix A, Tables A3 and A4.


Roots
Level 2
Level 1
Level 0
Document clusters defined by query-clustering
Documents
Level 3
Level 2
Level 1
Level 0
Document clusters formed by clustering higher level clusters

(a) Single-level

(b) Multiple-level

Single- and Multiple-Level Clustering

Figure 1

C) Phase 2: Clustering the Documents

In general, the most convenient way to implement phase 2 consists in using relevancy decisions. This is done by assigning each document to the cluster or clusters whose underlying query cluster(s) contain(s) one or more queries to which it is relevant. Unfortunately, this process deals with only part of the documents in the collection, in most cases. Although each document in the Cranfield 424 collection is relevant to one or more queries in that collection, not all of these queries are present in any of the cluster-sets CS1, CS2, and CS3. The documents not assigned to clusters by analysis of relevance may be called "loose documents" (see [8] for a full description of the term), and must be "blended" into the clusters already formed. In all three cases the number of loose documents is rather substantial: 135 in CS1, 89 in CS2, and 54 in CS3. (It should be noted here that, because of the relationship between these three cluster-sets, the loose documents of CS3 are a subset of those of CS2, which are, in turn, a subset of those of CS1.)

Because of the large numbers of loose documents, the method used to incorporate such documents into particular clusters is quite important, and must be the subject of careful scrutiny in any actual use of this method. For the present experiment, however, where the major object is the examination of the results of varying aspects of the clustering scheme other than the blending method, an arbitrary way of assigning loose documents to clusters is chosen. The method consists in correlating each loose document with the 15 centroids of the given cluster-set, and to add each document to the document cluster(s), for which the centroid of the underlying query cluster(s) satisfies one of the following conditions: either the correlation between query centroid and document is 0.250 or higher, or the centroid-document correlation is higher than the correlation with all other query centoids. The reason for this method will become clear when method 2b is discussed below. Results of this assignment — relevancy combined with blending — are given in Appendix A, Table A5.

It is this type of clustering which has been studied previously, and which seems most likely to produce improvements over standard algorithms which do not use query clusters. The effect is to classify documents according to the questions to which they relate, rather than according to similarities in word content. It is, moreover, this method which seems likely to exhibit the most improvement when a larger cluster-base is used. If such a trend could be confirmed, this type of query-clustering would produce a way for an information-retrieval system to "learn" from its past successes, while keeping it from repeating past mistakes. It might also be a way of implicitly altering a system to compensate for changes in terminology, or to anticipate the development of new fields of information. For these reasons, this form of query clustering may have the same type of advantages as document-space modification, a technique examined in Brauen [12].

Type 2 document clusters are formed according to correlations between documents and centroids of the clusters formed in phase 1. In some ways, this might be looked at as a standard clustering algorithm which begins with certain clusters already formed, and continues by blending into these the set of documents to be clustered. It might eventually be shown experimentally that using such centroids as a "seed collection" in any of the standard algorithms will produce improved performance from the final clusters.

In the experiment at hand, the procedure is the following for phase 2 using type 2 clusters: all 424 documents are correlated with all 15 query centroids in each cluster set. A document is assigned to any document cluster, for which the centroid of the underlying query cluster satisfies either of two conditions:

a) the centroid and the document correlete at a level of 0.250 or higher;

b) the correlation with the document is higher than that achieved by all other centorids.

Note that this is the same condition under which a loose document is blended into a cluster for method 1. This criterion is chosen, by inspection, to approximate the size and degree of overlap of what might be considered "standard clusters". While such an arbitrary cutoff value is likely to be used in any operational implementaion of this method, the cutoff may, of course, be varied to achieve different clusterings. In appendix A, Table A6 it may be noted that the sizes of the clusters formed vary greatly, from a minimum of 4 by CS1 to a maximum of 85 by CS3. This is clearly an undesirable result, and a method of avoiding it is suggested below. (Varying the cutoff might reduce the problem, but would probably not solve it entirely.)

It should be noted here that method 2 is unlike the previous one in that no loose documents result. This is due, of course, to taking each document individually and assinging it to one or more clusters. The problem of non-uniformly-sized clusters may be solved if the generation of loose documents is permitted. Instead of correlating documents against centroids, it is possible to reverse the process, matching centroids against documents. In this variation of method 2, the top n, say, correlants of each centroid are chosen for inclusion in the document cluster related to that centroid.

Thus, all clusters have the same number of items. By varying the cutoff (n), and imposing additional restrictions on correlations of included documents, it seems likely that interesting results could be obtained. This range of experimentation is not, however, included in the current study.

Finally, the third form of phase 2 is achieved here by correlating all of the 424 documents against the 75, 100, and 125 queries of the three cluster-sets. Documents are assigned to any document cluster whose underlying query cluster contains one or more queries such that either a) that query correlates more highly with that document than any other query, or b) the correlation between the query and the document is 0.250 or more. The motivation for this choice of method is the same as that for the type 2 method, and the same comments apply. In this application, also, a disparity arises in cluster sizes. Table A7 of Appendix A shows that cluster-set CS1 produces both the largest cluster (111 documents) and the smallest (3 documents) generated by method 3.

As before, a reverse strategy may be used which would ensure an even distribution of the documents throughout the clusters (aside from the blending of loose documents).

This method (in either variation) may be regarded as "pre-searching" the document collection in order to make later searches more effective. If, as assumed, many new queries are similar to queries already present in an information-retrieval system, then such a new query should easily find the cluster associated with these similar queries. This method of assigning documents to clusters thus guarantees that the documents in that cluster are those which correlate highly with the old, similar queries (and, thus, hopefully, with the new query).

Both of these last two implementations of phase 2 are inherently more expensive than the first. Relevant documents for a given already-processed query can be easily selected, without the necessity of correlating large numbers of concept vectors. In the case of method 2, each document must be correlated against as many centroids as there are query clusters. Method 3 requires a full search of all queries against all documents, although this would be done only once for each document and query. For method 2, a new correlation by all documents would be required with each update of the query clusters. Further analysis of comparative costs of these three methods is possible, but beyond the scope of this paper.

D) Phase 3: Assigning Centroids

The two methods of assigning centroids to document clusters are quite straightforward. In one case, the centroid is taken as that of the underlying query cluster. In the other, it is the standard centroid defined by the documents in the cluster.

Note that using document centroids generally requires an additional series of computations while the use of query centroids does not. This is the case because, as a rule, the process of query-clustering in phase 1 produces the query centroids as a side-effect, at no additional cost. In addition, query centroids tend to be small, taking up less storage space within the machine than document centroids. On the other hand, it may be that document centroids — which contain more of the information about the documents they represent — form a better vehicle for combining documents than query centroids. Even the best clusters will achieve poor performance if the centroids are poorly defined — see Section 4 of this paper — so that this choice, also, is crucial.

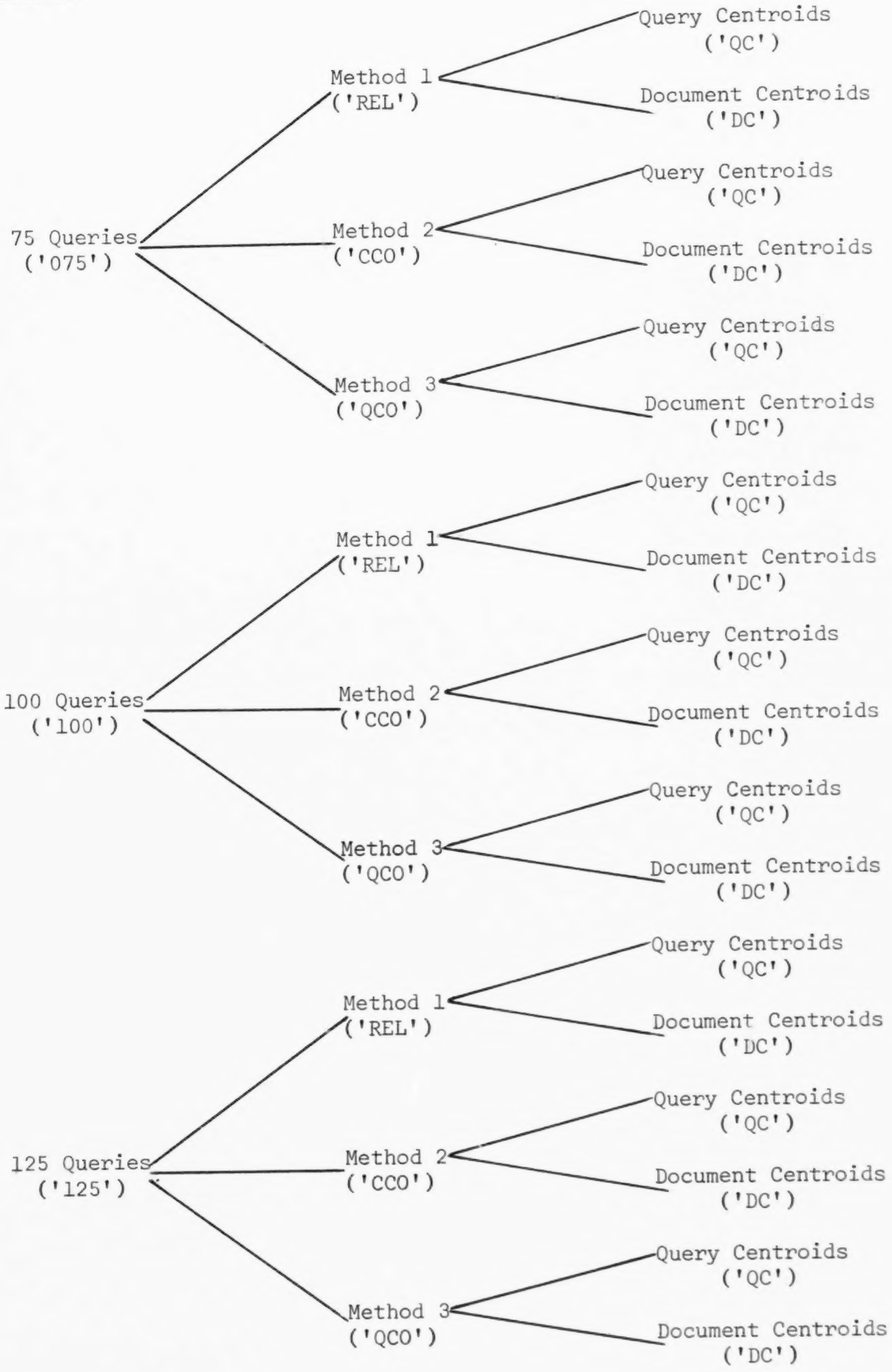
E) Summary

The diagram of Figure 2 indicates the variations used in forming 18 cluster trees from the Cranfield 424 collection. The cluster-collection names used in the rest of this paper may be drived from this diagram by concatenating the three keys describing the particular collection. For example, the collection formed from CS2 (100 queries) using documents assigned by method 2 (centroid correlations) and query centroids in phase 3 is '100CCOQC'.

3. Results

For the most part, the results of this experiment are unexpected. (Graphs of recall-precision values for all 18 clusters are given in Appendix B, together with graphs for the "standard" clusters and a full search using the test-set queries.) Consider, for example, the six comparisons which may be made varying only the number of queries clustered. (The list, including the names introduced in Figure 2, is given in Table 1 below.) Intuitively, the expected ranking in all cases is 125-100-075, best to worst. In only one case out of six, however, is this order achieved: clusters 125QCOQC, 100QCOQC, and 075QCOQC. (See Figure B7.) In four of the five other cases, the clusters using CS2 (100 queries) performed best. (In the remaining case, nnnRELQC, CS3 was best, but CS1 was second-best instead of last.) Moreover, only three of the eighteen cluster-sets produced better results than the "standard" clusters (clusters 100RELDC in Figure B2 and 100CCODC and 075CCODC in Figure B3.) Reference [2] predicts a different result.

The explanations for these results must await further analysis. At present, some observations may be given. It must be noted first that the


75 Queries
('075')
Method 1
('REL')
Query Centroids
('QC')
Document Centroids
('DC')
Method 2
('CCO')
Query Centroids
('QC')
Document Centroids
('DC')
Method 3
('QCO')
Query Centroids
('QC')
Document Centroids
('DC')
100 Queries
('100')
Method 1
('REL')
Query Centroids
('QC')
Document Centroids
('DC')
Method 2
('CCO')
Query Centroids
('QC')
Document Centroids
('DC')
Method 3
('QCO')
Query Centroids
('QC')
Document Centroids
('DC')
125 Queries
('125')
Method 1
('REL')
Query Centroids
('QC')
Document Centroids
('DC')
Method 2
('CCO')
Query Centroids
('QC')
Document Centroids
('DC')
Method 3
('QCO')
Query Centroids
('QC')
Document Centroids
('DC')

Formation of 18 Cluster Collections

Figure 2

1. nnnRELDC	2. nnnRELQC
075RELDC 100RELDC 125RELDC	075RELQC 100RELQC 125RELQC
3. nnnCCODC	4. nnnCCOQC
075CCODC 100CCODC 125CCODC	075CCOQC 100CCOQC 125CCOQC
5. nnnQCODC	6. nnnQCOQC
075QCODC 100QCODC 125QCODC	075QCOQC 100QCOQC 125QCOQC

Six Comparisons of Cluster Groups
by Number of Queries Clustered

Table 1

decisions on the rankings of two or more search results is being made on the basis only of cursory analysis of the recall-precision graphs presented in Appendix B. On that basis, for example, clusters 100RELDC (Figure B2) are being called "better" than the standard clusters of Figure B1, even though the latter rises above the former from recall level .35 onward. A more thorough investigation of these graphs is needed to reach firm conclusions concerning the preferred clustering method.

The possibilities of experimental errors must also be considered. In Appendix C the procedures used in setting up all of these collections are described. Because of the large amount of handwork that went into this stage of the experiment, and because of the lack of complete verification of the resulting lists, it is possible that some degree of error has been introduced in this way.

The final point to be mentioned here concerns the search strategy used. In [2], it is necessary to compensate for unusually large clusters by varying the number of clusters expanded during searching. Since this is not a problem in the present experiment, a fixed number of clusters is expanded in all searches. It is possible that this number does not allow proper representation of the properties of the existing collections. Future searches will include fixed expansions of different values, along with expansions which vary with correlation and number of documents searched. This last consideration seems most promising as an explanation of the results reported here.

4. Principles of Evaluation

In [2] it is suggested that different types of evaluation parameters are needed in full- and clustered-searches. In particular, in a clustered search, it is possible to isolate the cause of good (or bad) performance to one of three areas:

1) Cluster generation

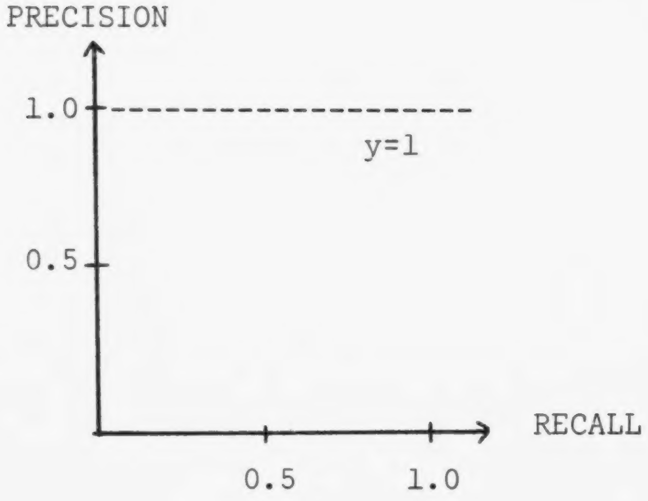
2) Centroid definition

3) Search strategy.

When the search is done without clustering, only the third explanation applies. In addition, the amount of work done in searching a clustered collection may vary from query to query and from one implementation to another, and should also be measured. Although such a measure is sometimes included in the basic performance of a system, it seems clear that the two areas of analysis are quite separate and should be treated as such.

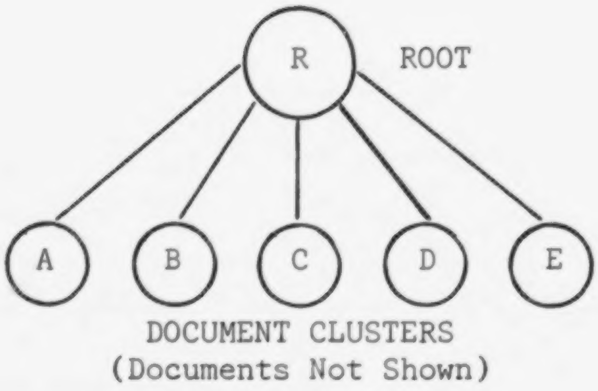
In general, search results are compared to an "optimal" system: one which produces all relevant documents for all queries ranked at the top of the list of retrieved items. In such a system the recall-precision graph achieves a horizontal line at y-intercept 1 (see Figure 3). In the same way, it is psssible to rank the first-level clusters in any clustered collection for each query, in order of "desirability". (Note that the analysis which follows is not directly applicable to multiple-level clusterings. This problem is discussed in [10] and [13].)

Given the configuration of Figure 4, the hoped-for ordering of these clusters before expansion is obviously (high to low number of relevant documents) D-C-A-B-E. Note that two considerations are involved here: First, it is required that the clusters containing the greatest number of relevant documents be ranked highest. Second, between two clusters whose contents of relevant documents are the same, the smaller should be first. The definition of the centroid for each cluster determines how high each cluster ranks, and that definition will determine whether a search does



Optimal Recall-Precision Graph

Figure 3


R
ROOT
A
B
C
D
E
DOCUMENT CLUSTERS
(Documents Not Shown)

CLUSTER	NO. RELEVANT TO p	NO. IN CLUSTER
A	3	40
B	1	10
C	5	30
D	15	50
E	1	15

n = total documents relevant to p = 25

Clusters with Relevant Documents for Query p

Figure 4

well (or poorly) because the proper clusters are (or are not) expanded.

Consider now a search which orders the clusters of Figure 4 in their "optimal" ranking. This search will, in general, perform less well than a similar one with the clusters of Figure 5, where the clusters are also considered to be ordered correctly before expansion. The reason is clear: in the former case it is necessary to examine 135 documents before all relevant may be included, while with the second group only 80 document correlations must be made. The cluster-set exhibited in the second set possesses as few nonrelevant as possible for those clusters containing all relevant. This property is determined by cluster generation, as apart from centroid generation and search strategy.

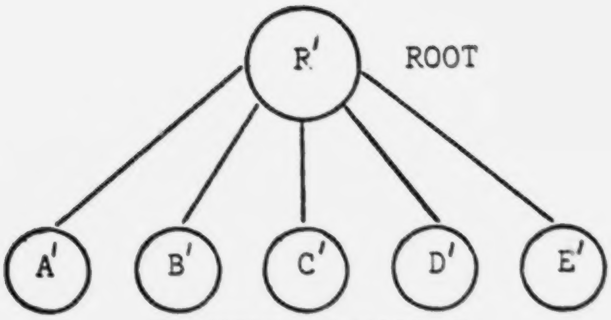
Three parameters are introduced in [2] for dealing with these concepts. They are "aim", "target", and "rejection", defined as follows: Given a query $\underline{q}$ with $\underline{n}$ relevant documents. It is assumed that a clustered document collection is to be evaluated according to its clustering success and its achievement in centroid definition. For each number $\underline{c}$ of clusters expanded,

a) the aim clusters are those $\underline{c}$ clusters ranked first by whatever correlation procedure is used in ranking clusters, and

b) the target clusters are those $\underline{c}$ clusters ranked first according to the previously-discussed considerations of a number of relevant contained and size. (For a more precise description, including the question of how documents appearing in more than one cluster are to be treated, see [2].)

The aim value is defined as

$$\frac{\text{number of relevant documents in aim clusters}}{\underline{n}}$$



DOCUMENT CLUSTERS
(Documents Not Shown)

CLUSTER	NO. RELEVANT TO p	NO. IN CLUSTER
A'	0	40
B'	0	10
C'	0	30
D'	25	80
E'	0	15

n = total documents relevant to p = 25

Clusters with Relevant Documents for Query p
(Second Set)

Figure 5

and the *target value* is

$$\frac{\text{number of relevant documents in target clusters}}{n}.$$

Similarly, *rejection* is defined as

$$\frac{\text{occurrences of rel. documents in target clusters}}{\text{occurrences of rel. documents in all clusters}}.$$

In the measures above, optimal performance is achieved when both the *aim-to-target-ratio* (self-defining) and the rejection are 1 when averaged over all queries. This is a restatement of the definitions of [2], in a more precise form.

The recall-precision graphs are included in Appendix B, and an explanation of the programming tasks is given in Appendix C.

References

[1] V. R. Lesser, A Modified Two-Level Search Algorithm Using Request Clustering, Report No. ISR-11 to the National Science Foundation, Section VII, Department of Computer Science, Cornell University, June 1966.

[2] S. Worona, Query Clustering in a Large Document Collection, Report No. ISR-16 to the National Science Foundation, Section XV, Department of Computer Science, Cornell University, September 1969.

[3] J. D. Broffitt, H. L. Morgan, and J. V. Soden, On Some Clustering Techniques for Information Retrieval, Report No. ISR-11 to the National Science Foundation, Department of Computer Science, Cornell University, June 1966.

[4] R. T. Grauer and M. Messier, An Evaluation of Rocchio's Clustering Algorithm, Report No. ISR-12 to the National Science Foundation, Section VI, Department of Computer Science, Cornell University, June 1967.

[5] R. T. Dattola, A Fast Algorithm for Automatic Classification, Report No. ISR-14 to the National Science Foundation, Section V Department of Computer Science, Cornell University, October 1968.

[6] S. Rieber and V. P. Marathe, The Single Pass Clustering Method, Report No. ISR-16 to the National Science Foundation, Section XIV, Department of Computer Science, Cornell University, September 1969.

[7] R. T. Dattola, Experiments with a Fast Algorithm for Automatic Classification, Report No. ISR-16 to the National Science Foundation, Section XIII, Department of Computer Science, Cornell University, September 1969.

[8] J. J. Rocchio, Document Retrieval Systems — Optimization and Evaluation, Report No. ISR-10 to the National Science Foundation, Harvard University, March 1966.

[9] G. Salton, Automatic Information Organization and Retrieval, McGraw-Hill, Inc., New York, 1968, Ch. 4.

[10] D. Murray, unpublished manuscript.

[11] R. Magliozzo and M. Bodenstein, unpublished manuscript.

[12] R. Brauen, Document Vector Modification, Report No. ISR-17 to the National Science Foundation, Department of Computer Science, Cornell University, September 1969.

[13] D. Murray, On the Optimal Ranking of Clusters, unpublished manuscript.

[14] D. Williamson, R. Williamson, and M. Lesk, The Cornell Implementation of the SMART System, Report No. ISR-16 to the National Science Foundation, Section I, Department of Computer Science, Cornell University, September 1969.

Appendix A

Tables of Statistics

Query Number	Cran4 Number	Cran14 Number	Author
1	17	67	107
2	18	68	107
3	19	69	107
4	31	97	95
5	32	98	95
6	33	99	13
7	34	100	13
8	43	110	80
9	44	111	80
10	45	112	80
11	53	122	25
12	54	123	25
13	60	137	46
14	61	138	46
15	62	139	46
16	64	145	83
17	65	146	83
18	71	157	17
19	72	158	17
20	81	175	32
21	82	176	32
22	84	183	165
23	85	184	165
24	94	206	147
25	104	226	47
26	123	274	165
27	128	291	17
28	131	295	38
29	132	296	38
30	136	301	32

"Cran4 Number" is the query's number in the 424 collection.
"Cran14 Number" is the query's number in the 1400 collection.
"Author" is the author-number, as given by Cranfield.

Test-Set Queries

Table A1

Qu No.	Clus Set	C4 No.	C14 No.	Qu No.	Clus Set	C4 No.	C14 No.	Qu No.	Clus Set	C4 No.	C14 No.
1	CS1	1	1	43	CS1	99	215	85	CS2	50	119
2	"	2	2	44	"	100	217	86	"	56	131
3	"	3	9	45	"	103	225	87	"	57	132
4	"	6	15	46	"	105	230	88	"	58	135
5	"	7	18	47	"	106	231	89	"	67	148
6	"	10	34	48	"	108	233	90	"	78	148
7	"	12	41	49	"	109	245	91	"	87	190
8	"	13	51	50	"	110	246	92	"	93	205
9	"	14	58	51	"	111	247	93	"	101	218
10	"	15	59	52	"	113	252	94	"	102	224
11	"	21	74	53	"	115	259	95	"	122	273
12	"	23	80	54	"	117	265	96	"	127	288
13	"	25	82	55	"	118	266	97	"	135	299
14	"	28	87	56	"	120	269	98	"	143	332
15	"	29	94	57	"	121	272	99	"	145	335
16	"	30	95	58	"	124	283	100	"	150	352
17	"	37	103	59	"	125	284	101	CS3	5	13
18	"	38	104	60	"	126	285	102	"	8	26
19	"	40	107	61	"	129	293	103	"	24	81
20	"	47	114	62	"	133	297	104	"	27	85
21	"	49	118	63	"	134	298	105	"	35	101
22	"	51	120	64	"	137	306	106	"	39	106
23	"	52	121	65	"	138	314	107	"	41	108
24	"	55	130	66	"	139	315	108	"	42	109
25	"	59	136	67	"	140	316	109	"	48	116
26	"	63	142	68	"	141	321	110	"	70	156
27	"	66	127	69	"	142	323	111	"	73	160
28	"	68	152	70	"	146	336	112	"	79	169
29	"	69	155	71	"	147	347	113	"	83	182
30	"	74	161	72	"	148	348	114	"	89	200
31	"	75	163	73	"	153	356	115	"	90	201
32	"	76	165	74	"	154	360	116	"	107	232
33	"	77	167	75	"	155	365	117	"	112	250
34	"	80	171	76	CS2	4	12	118	"	114	255
35	"	86	187	77	"	9	33	119	"	116	261
36	"	88	196	78	"	11	39	120	"	119	268
37	"	91	202	79	"	16	66	121	"	130	294
38	"	92	204	80	"	20	72	122	"	144	333
39	"	95	209	81	"	22	79	123	"	149	349
40	"	96	210	82	"	26	83	124	"	151	353
41	"	97	211	83	"	36	102	125	"	152	355
42	"	98	214	84	"	46	113				

"C4 No." is the query number in the 424 collection.
"C14 No." is the query number in the 1400 collection.
"CS1", "CS2", and "CS3" mark the beginning of the additional queries for the collections. CS1 consists of queries 1-75; CS2 consists of 1-100; CS3 consists of 1-125.

Cluster-Set Queries

Table A2

Collection Name	CS1	CS2	CS3
No. of queries in collection	75	100	125
Number of clusters formed	15	15	15
Items in largest cluster (no.)	9 (1)	11 (6)	14 (5,10)
% of collection in largest cl.	11	11	12
Items in smallest cluster (no.)	2 (10,12)	3 (8,13,14)	4 (8,11,12)
% of collection in smallest cl.	3	3	3
Repetition factor*	0	0	0

*Defined as the number of total occurrences of documents or queries throughout a clustered collection, divided by the number of distinct items in the collection.

Results of Query Clustering

Table A3

Collection Name	Cran 424 Docs
No. of documents in collection	424
Number of clusters formed	15
Items in largest cluster (no.)	76 (4)
% of collection in largest cl.	18
Items in smallest cluster (no.)	15 (9)
% of collection in smallest cl.	4
Repetition factor*	285

*See note on Table A3.

Results of Clustering the Cranfield 424 Document Collection Using Dattola's Algorithm

Table A4

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	82 (1)	85 (6)	78 (9)
% of collection in largest cl.	19	20	18
Items in smallest cluster (no.)	5 (10)	14 (13,14)	22 (12)
% of collection in smallest cl.	1	3	5
Repetition factor*	113	178	272

*See note on Table A3.

Results of Type 1 Clustering
(nnnRELxx)

Table A5

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	80 (2)	81 (15)	85 (14)
% of collection in largest cl.	19	19	20
Items in smallest cluster (no.)	4 (15)	6 (13)	11 (8)
% of collection in smallest cl.	1	1	3
Repetition factor*	64	76	52

*See note on Table A3.

Results of Type 2 Clustering
(nnnCCOxx)

Table A6

Collection Prefix	075	100	125
No. of documents in collection	424	424	424
Number of clusters formed	15	15	15
Items in largest cluster (no.)	111 (1)	107 (6)	89 (14)
% of collection in largest cl.	26	25	21
Items in smallest cluster (no.)	3 (15)	7 (13)	19 (8)
% of collection in smallest cl.	1	2	4
Repetition factor*	171	295	388

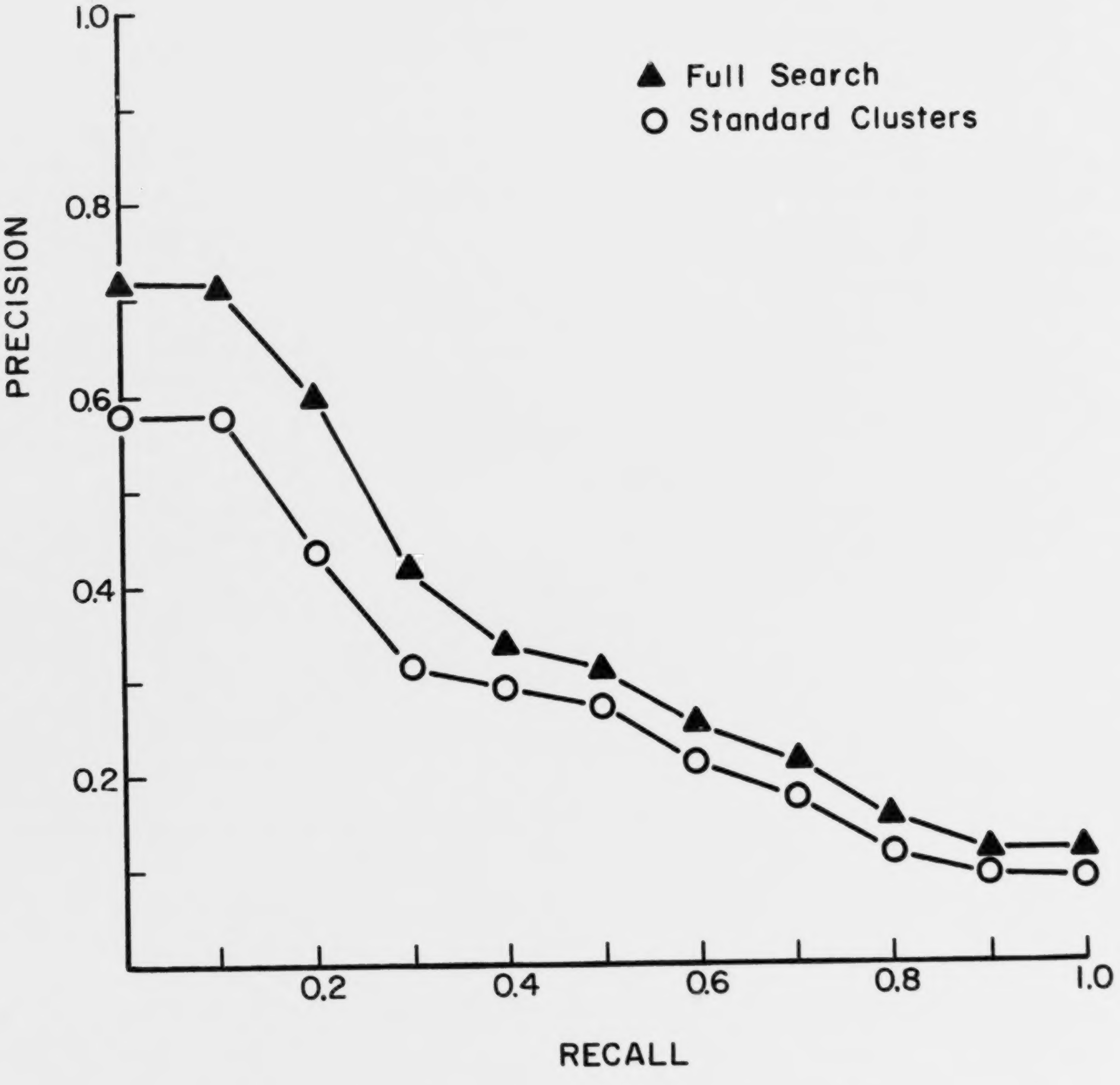
*See note on Table A3.

Results of Type 3 Clustering
(nnnQCOxx)

Table A7

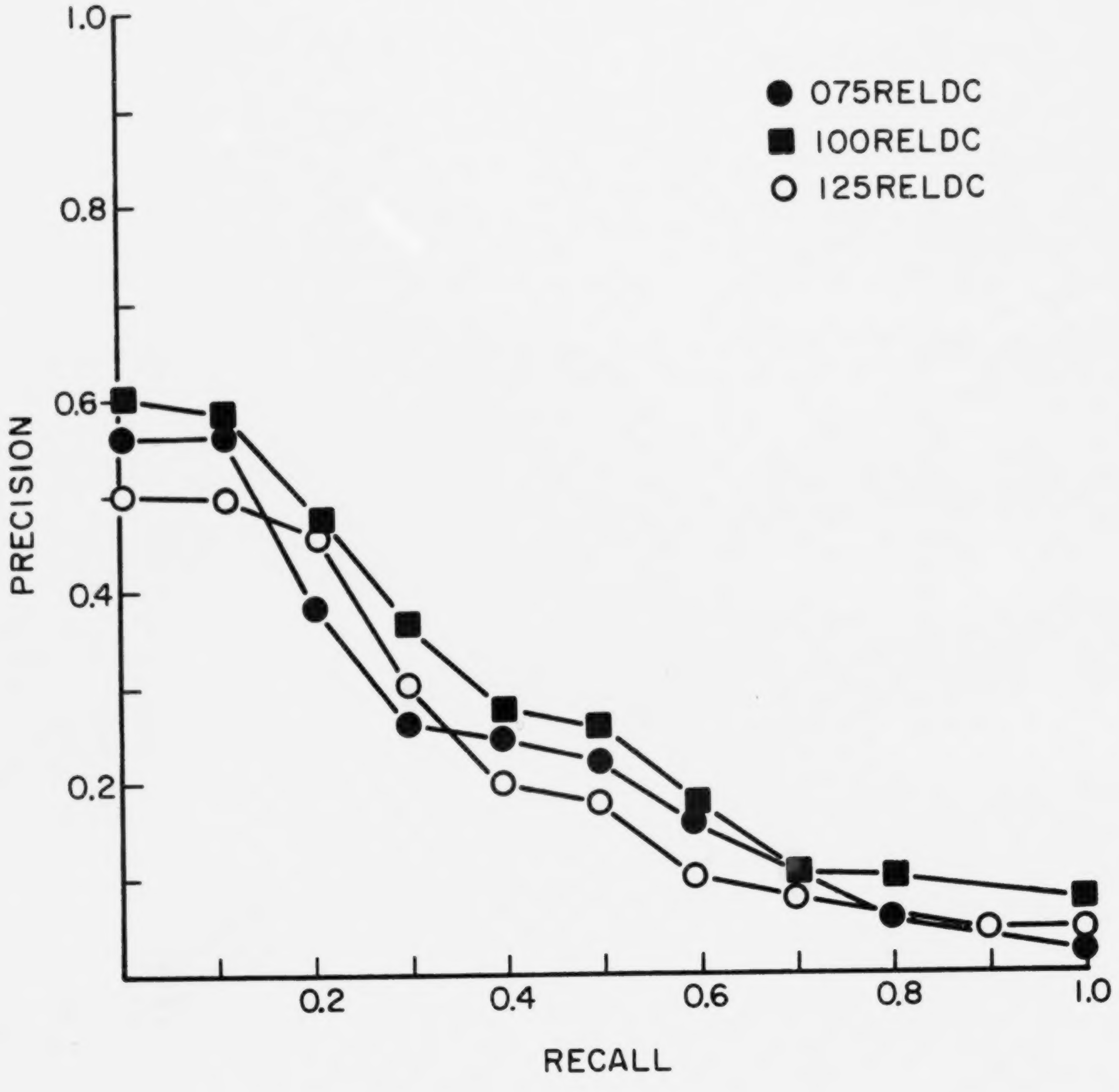
Appendix B

Recall-Precision Graphs of Results



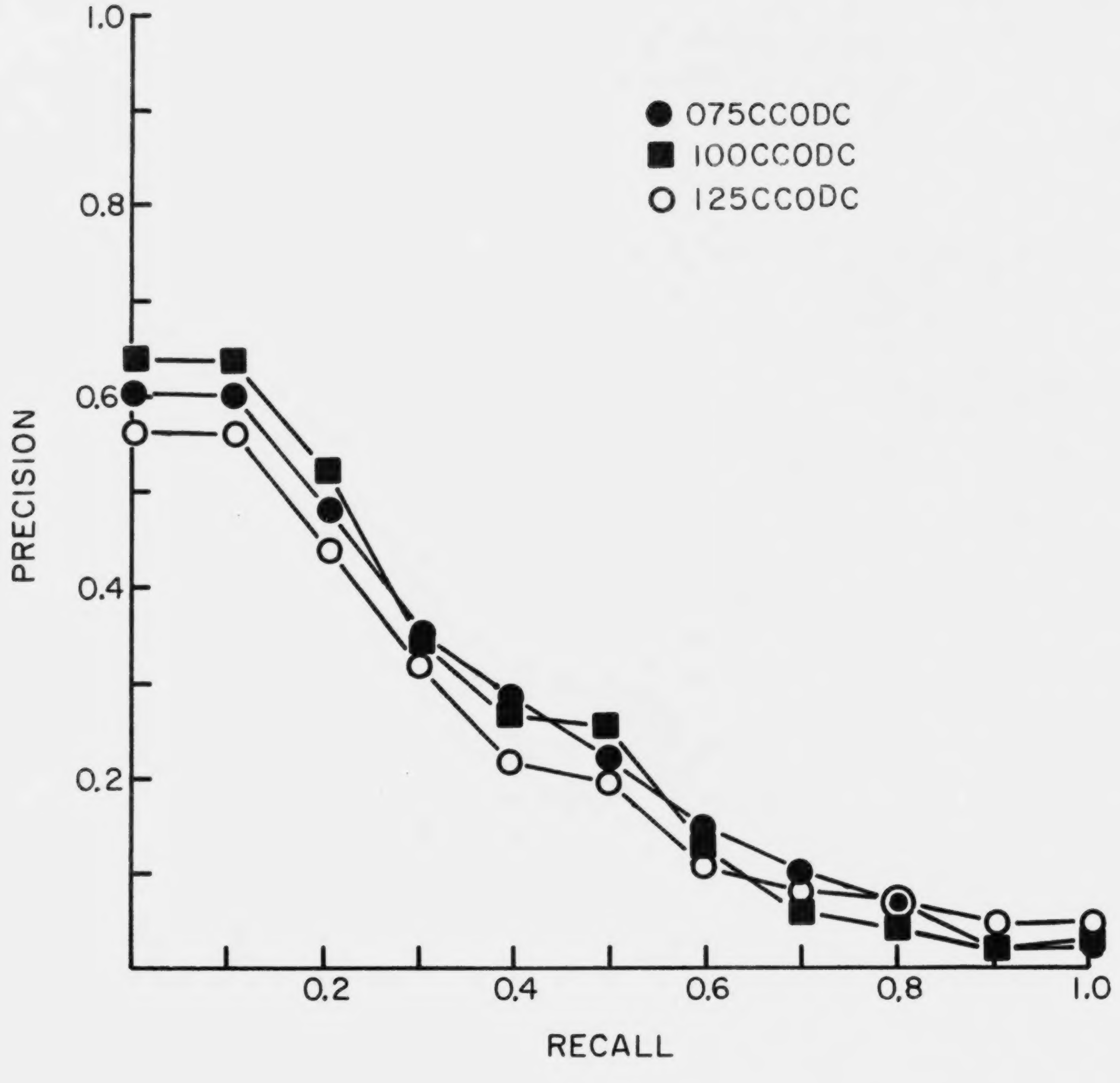
Full Search and Standard Clusters

Figure B1



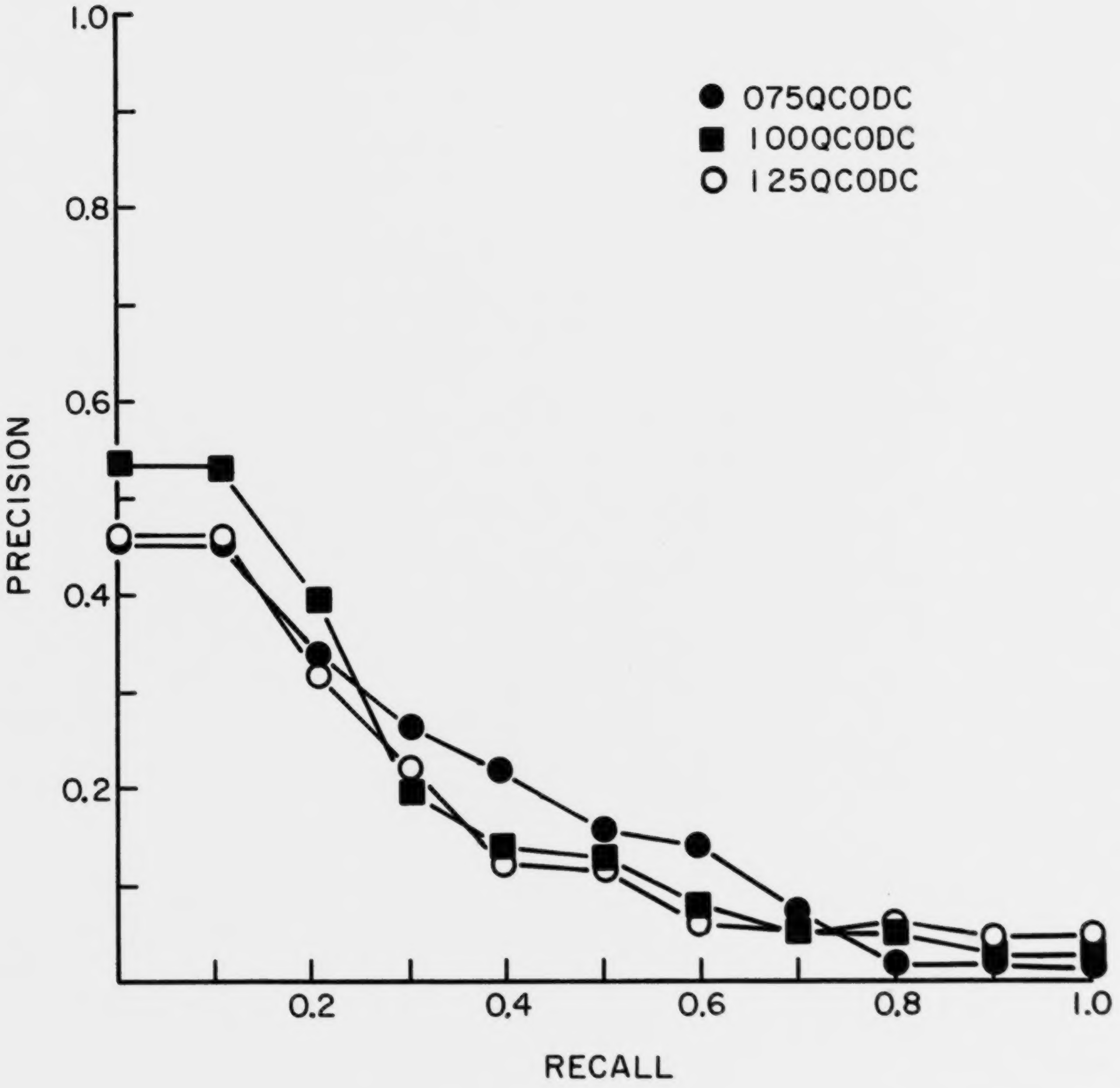
nnnRELDC

Figure B2



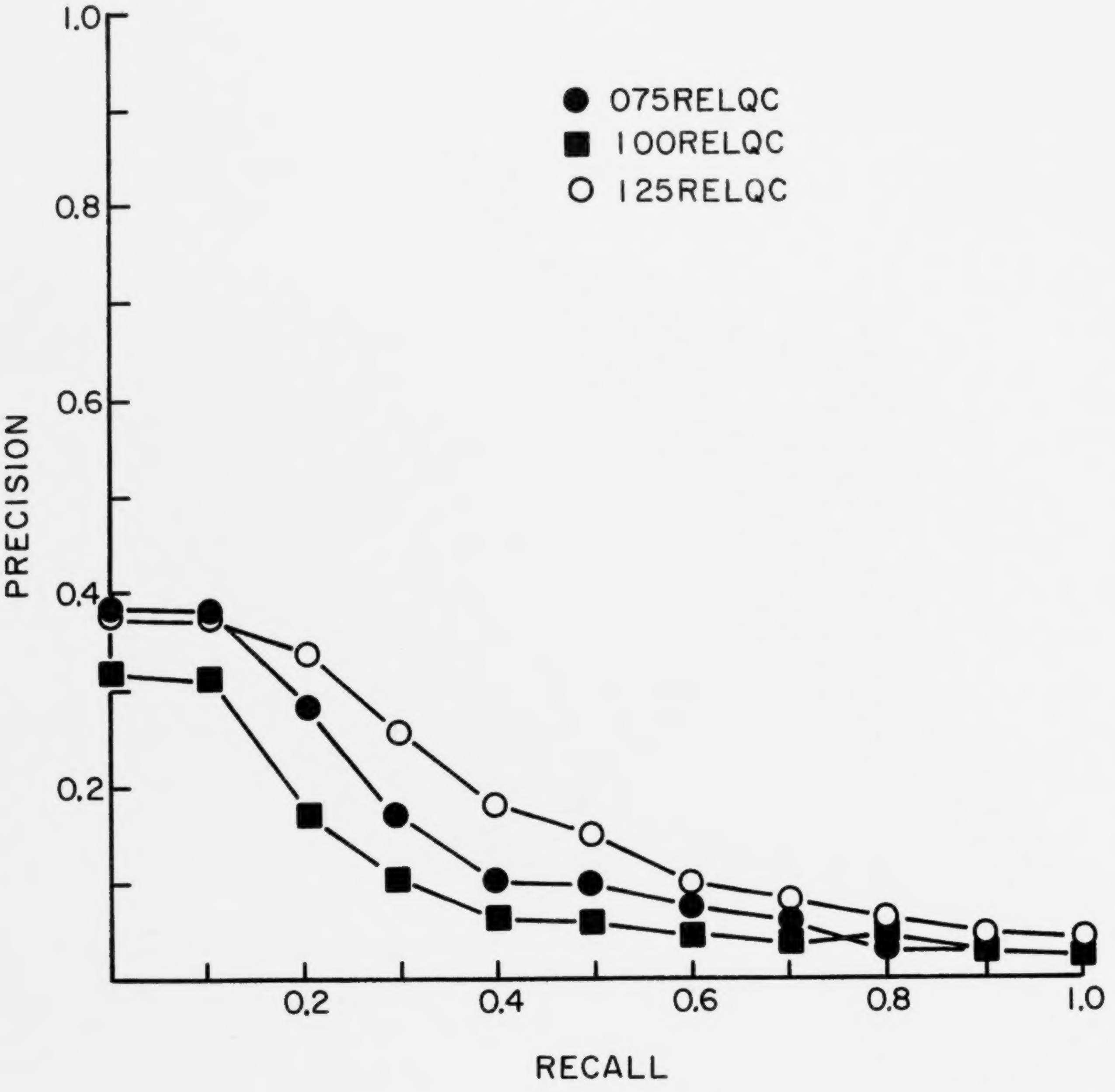
nnnCCODC

Figure B3



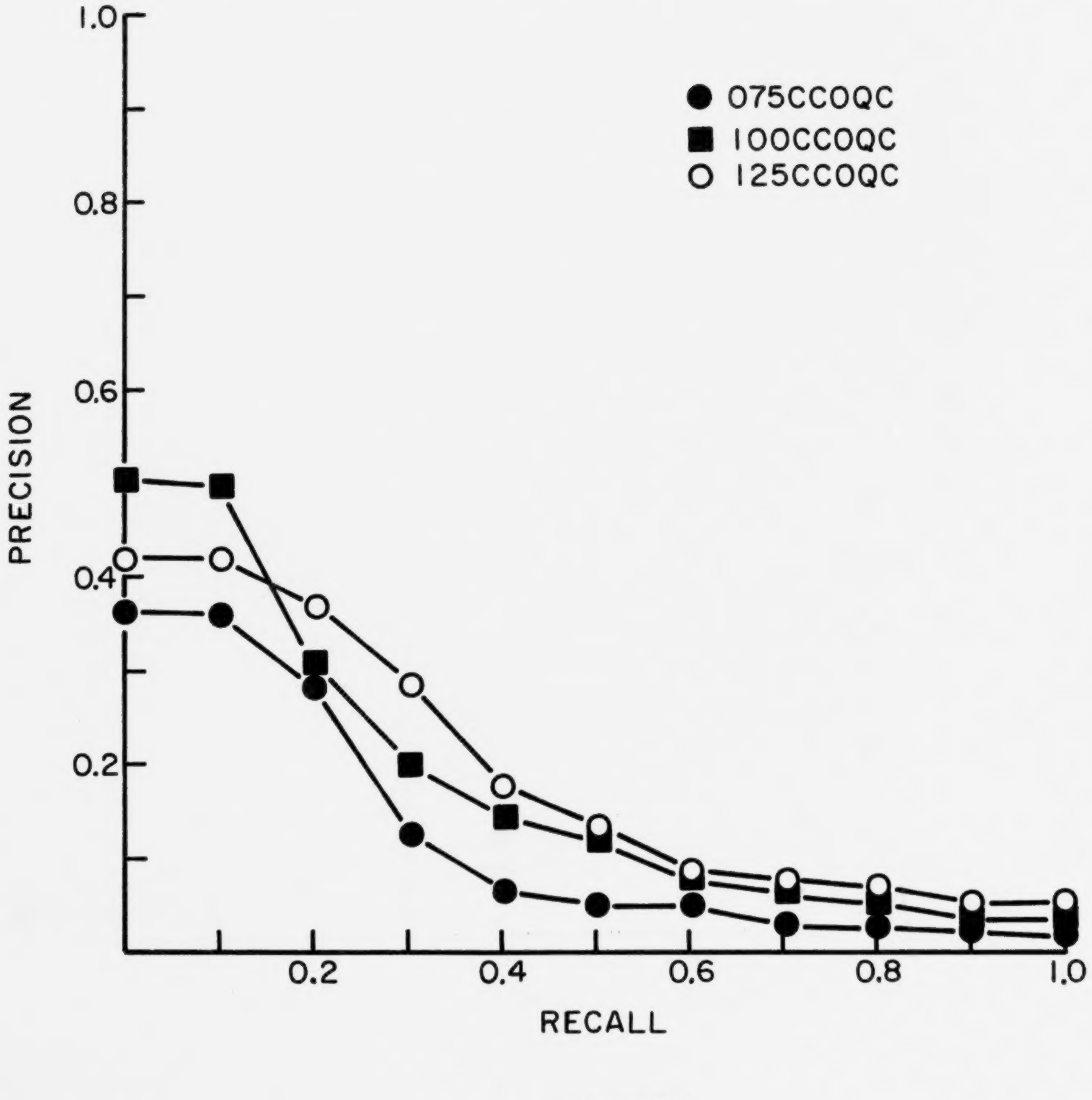
nnnQCODC

Figure B4



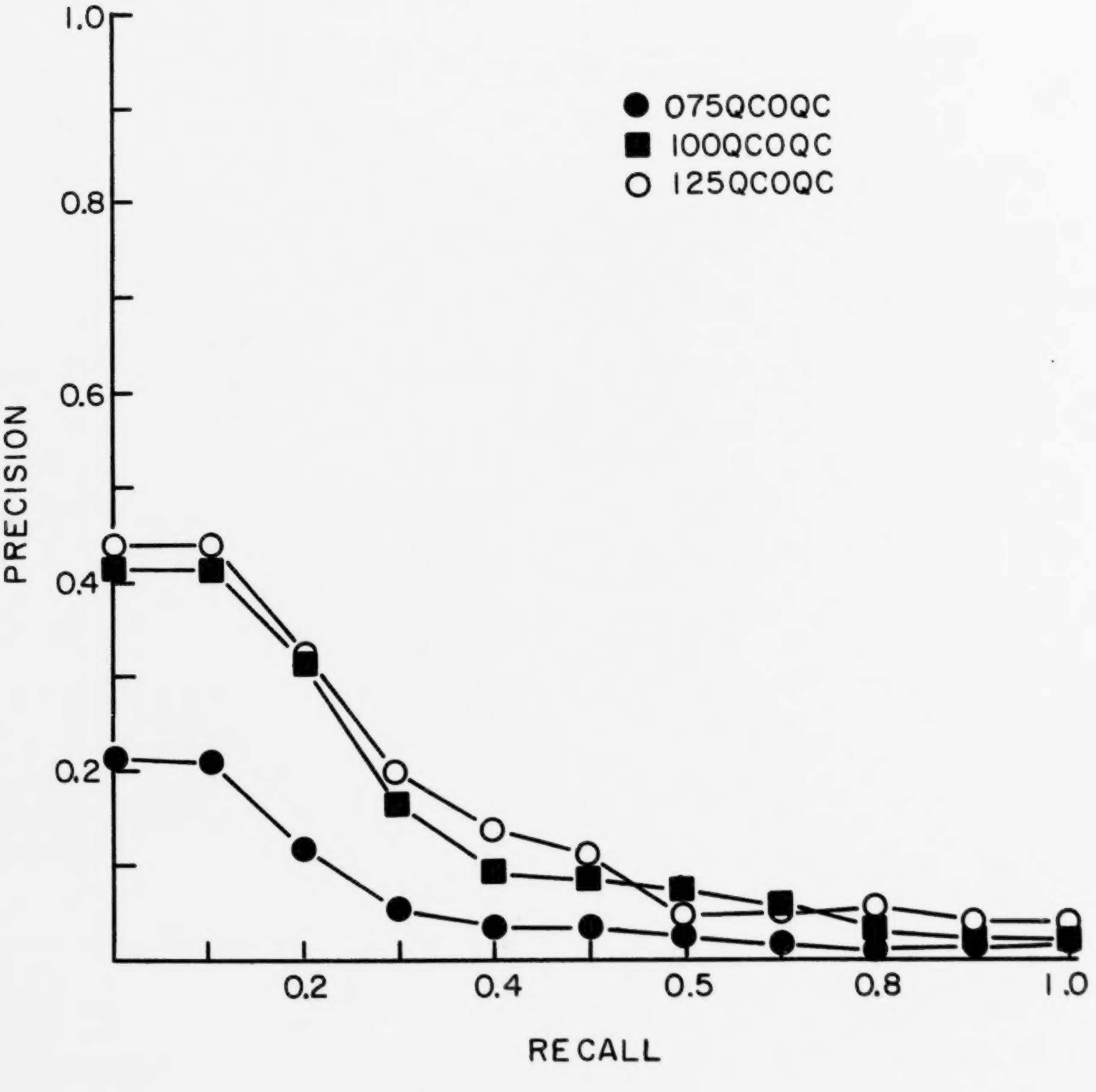
nnnRELQC

Figure B5



nnnCCOQC

Figure B6



nnnQCOCQ

Figure B7

Appendix C

The SMART System

In [14] the basic facts about the SMART system implemented at Cornell University are given. At the time of its design, however, no large-scale use of query clustering had been attempted, and none was planned. Therefore, the experiment described in this report required some ad hoc programming and some tiresome hand work. It is hoped that future experimentors with query clustering will benefit from the description given here of the procedures necessary to work within this implementation of SMART in order to carry out such work.

No provision exists in SMART for performing the type of random-number generation and author-set-maintenance described in Section 2. (This is by no means a deficiency of the implementation since such a program is of little general use.) A program was therefore written in FORTRAN to take the information about authors of queries and produce a randomly-selected set subject to the constraints mentioned previously. Such author information is readily available.

At the beginning of the experiment no SMART procedure existed for forming a subcollection of a query or document collection included within the system. Another program was therefore written (also in FORTRAN) to subdivide the Cranfield 424 collection's queries into the four subsets necessary for the experiment (the test-set, and cluster-sets CS1, CS2, and CS3). Recently, however, D. M. Murray has written an addition to the SMART system which performs the necessary subsetting within SMART.

After creating the cluster-sets CS1, CS2, and CS3, Dattola's algorithm (implemented by the SMART procedure DCLSTR) was applied to these collections.

At the same time, the Cranfield 424 documents were processed by DCLSTR. As explained in the text, exactly 15 clusters were required in all cases. It is a property of Dattola's algorithm that the number of clusters produced at any time is a function of the collection used, the random seed specified, and the number of clusters requested. It is thus not possible to predict with accuracy how many clusters will be produced from any set of these parameters. To obtain exactly 15 clusters of the Cranfield documents it was necessary to use several attempts: cluster-set CS2 required 4 tries, cluster-set CS3 needed 7, while only CS1 was successfully divided into 15 clusters in just 1 attempt. A summary of these trial-and-error processes is given in Table C1.

Method 1 of phase 2 was implemented by keypunching the numbers of the documents relevant to each of the queries in each cluster of CS1, CS2, and CS3, and then sorting these three lists with another specially-written (although trivial) program, yielding a listing of the "non-loose" documents. The corresponding loose documents were listed by hand. It was originally assumed that methods 2 and 3 would be done by simple SMART searches, by using the 424 documents as queries against the three sets of centroids and the three sets of queries. This large number of "queries" proved unworkable in the system, and another program modification was required.

After the definitions for all 9 cluster sets were completed, it remained to generate 18 centroid sets, and unite these two parts of the ultimate collections. A SMART routine called CRDCEN has as its purpose this exact function. The experiment was delayed, however, by the necessity to keypunch a great deal of information from the previous tabulations. In it recommended that when a program is written to perform

Cluster Set	Attempt	Seed	No. Clusters Requested	No. Clusters Received
CS1	1	.12345	15	15*
CS2	1	.12345	15	12
	2	.54321	15	13
	3	.54321	16	14
	4	.54321	17	15*
CS3	1	.12345	15	14
	2	.54321	15	13
	3	.12345	16	14
	4	.12345	17	14
	5	.12345	18	16
	6	.54321	18	16
	7	.54321	17	15*
424 docs	1	.12345	15	11
	2	.12345	18	15*

*satisfactory clusters

Parameters Used in Generating Clusters
with Dattola's Algorithm

Table C1

the required tabulation automatically, this should be done with a view to obtaining the data and format required by CRDCEN, the process to which the results will eventually be passed.

Eventaully, the entire process should be made a part of the SMART system to be invoked like any standard clustering algorithm.

XIV. A Prototype On-Line Document Retrieval System

D. Williamson and R. Williamson

Abstract

A design is outlined for a SMART on-line document retrieval system, using console initiated search and retrieval procedures. The conversational system is described as well as the program organization.

1. Introduction

The SMART system presently contains routines for experimental, off-line document retrieval. The experimental results obtained so far indicate that automatic document retrieval can provide useful information for general library users. The next logical step is the development of a suitable user-oriented interface providing access via on-line consoles in an interactive manner.

This report describes a prototype, on-line document retrieval system and a user interface. The system which is outlined is intended to provide the best service possible to on-line users at a reasonable cost, but could also be efficiently used with very few modifications as a batch or remote entry system. While initial testing with collections of only a few thousand documents and less than five consoles is anticipated, the mechanisms used are intended to be applicable without revision to much larger collections of about 500,000 documents, and up to one to two hundred input-output consoles.

2. Anticipated Computer Configuration

In order to provide adequate response times — about 10 seconds for minor inputs and about 30 seconds for responses to search commands — a large, high-speed computer is necessary. Document retrieval, like many other non-numeric processes, requires a large data base of which a small, but substantial, fraction must be accessed for each query. Thus, it is necessary to operate with large, on-line files — presumably on a disk (although certain files could be placed on a data cell type device).

While a large computer is necessary to support the input-output equipment, and provide reasonable response times, an on-line retrieval system such as SMART, will not be able to utilize the full resources of a large machine. First, periods will occur when no users wish to avail themselves of the on-line system; and even when actual users are present, most of the real-time of an interaction is spent waiting for user decisions. Also, while processing a search request, the computer may be expected to be input-output (I-O) bound waiting for vocabularies and documents to be brought into core.

If processing costs are to be reasonable, provision must be made to permit non-retrieval users to process while the retrieval system is inactive for one reason or another. The type of environment needed is typified by many of the multi-processing and time-sharing systems available on large machines today. With these systems, jobs are effectively allocated to two queues: most are awaiting execution, and a few are in execution. Those in execution share the central processor (C.P.U.), memory, and on-line storage devices. Each memory area and storage device is usually dedicated to a single job. (In addition, a few devices and storage areas are normally reserved for the supervisor which is used by all jobs.) CPU allocation is normally switched from one executing job to another (through the supervisor) whenever that job is blocked — usually because it must await completion of an I-O transmission. System blocks are provided to prevent jobs from monopolizing the CPU, when no blocks occur for a certain time.

In the normal course of events, each executing job receives the opportunity to use the CPU several times a minute. Much of the time, a retrieval process such as SMART will be unable to utilize the opportunity to process. However when SMART has work to do, and the information necessary to do that work is available, the CPU is normally accessible — effectively instantaneously. The reason is that the retrieval tasks will appear as highly I-O bound jobs, which are therefore core resident for long periods of time, and are usually high in priority for CPU access.

SMART can make efficient use of as much core storage as can be made available. However, the retrieval routines tend to be small, and are highly overlayable; thus, the basic core area requirements are quite small. As in other typical data processing applications, the major core requirements in a retrieval program are for data areas in which to place I-O buffers for dictionaries, documents, etc. It would be most desirable if SMART could obtain 100 K to 200 K of core (possibly from a bulk core rather than from the high speed main core) on demand, for periods of only several seconds each time a request (or group of pseudo-batched requests) are processed. This core could easily come from the system buffer pool. However, sharing of core in this way is not a normal feature of today's operating systems; thus, SMART will undoubtedly have to reserve an area of high-speed core for programs (25-30 K bytes), and an area of bulk core for data (at least 50 K bytes — however, the more core is available, the faster will normally be the obtainable response times).

3. On-Line Document Retrieval — A User's View

When control of a console is transferred to SMART, the remote unit should be titled clearly to indicate to the user what basic information is needed at each step (detailed information should be provided as specified by a user's manual).

If SMART is on-line at the time of console transfer, the user must first enter such basic information as his name and account number (see Fig. 1). After this information is accepted by SMART, the user can proceed to ask for the execution of a given process. Many processes, such as query searches, query updates, and displays of output are available.

An initial user will probably start with a single query search (such as shown in Fig. 1). In this case, he will type in his query and then ask for a search to be done. The results will be displayed (in one of several possible forms, such as titles, abstracts, etc.), and the user will then either get a further display of the documents, or use the results of the search at that point.

Several types of display for retrieved documents could be used. The volume of information included in abstracts (or full articles) is likely to be so large that teletype display will be impractically slow; cathode-ray tube display is however quite expensive. Storage of abstracts at the remote terminal is an attractive alternative, with storage either on microfiche cards or in computer listings.

Following the retrieval of an initial set of abstracts, the query

```
$PROCEED
-SMART
#
#         SMART
#
#What is your name?-
-Joe Cornell
#What is your access code?-
-NONE
#Your access code is "MNAIZ".
#Do you wish to enter a query?
-Yes.
#Please enter your 1 th query.
#Type "End of query." when finished.
-What articles are there in ...
-
-                       .  End of query.
#Is your query ready for analysis?-
-Yes.
```

A Typical User's First Query

Fig. 1

Rank	Article	Correlation
1	60x1212 L. B. Heilprin, Towards a Definition of Information Science	0.6708
2	45x1215 D. Crosland, Graduate Training in Information Science	0.4472
3	03x1210 R. L. Taylor, In Information Science Education	0.3828
4	21x1209 Personnel — An Assessment and Projection	0.3660
5	43x1206 A. M. Rees, The Education of Science Information Personnel — A Challenge to the Library Schools	0.3651

Results of Initial Search of Query 1

Fig. 1 (continued)

Following the retrieval of an initial set of abstracts, the query author can return to the console and give the system his estimated relevance decisions. Since a prime source of error in all document retrieval systems is the discrepancy between a query author's intended query and his expressed query, initial queries can often be greatly improved through a process known as relevance feedback. This process modifies the query by adding words used in the relevant documents to the query, thus enlarging, and hopefully, improving the query. To improve his query, the user would re-enter the system, asking for a search on the original query plus relevant documents. An example of the re-entry to use feedback is shown in Fig. 2. In this case, the user asks to delete titles and uses only minimal replies. After the preliminary sign on at the console, the user is asked if he wishes to submit relevancy decisions for any active queries (in this case query 10). An indication must then be given of these decisions on a relevance scale from 1 to 5. After entering the decisions, the user asks for relevance feedback, and gets the results in a manner similar to the search results in Fig. 1.

For more experienced users, other procedures might be useful. Dictionary display to help the user construct more reasonable queries is possible, and various types of syntactic analysis can be used. The user can also alter the searching methods used by utilizing his private search parameters instead of the standard system parameters.

Each of the various procedures available to users requires specific patterns of interaction between the console and the user. Table 1 contains a tabular display of portions of a proposed console interface. Only a few of the procedures are traced in full, as an example of how such an interface would be constructed. The importance of the table lies in its overall structure — the specific wording of the messages and the division of labor among table segments is of minor interest. However, it should be noted that console interaction is handled in a sequential manner. Thus each user is associated with just one pointer indicating the segment to which he is replying.

Each table segment consists of one computer to console message including a possible user response, or system action. If an unanticipated response is obtained in a basic system, the text will be repeated in tutorial mode. In a more advanced system, special segments could be set up to handle unanticipated responses in special ways.

Several responses are global in that they could appear at any time rather than in response to a specific SMART message. These are listed in Table 1 under segment 0 (e.g. reply class shifts). The normal form of a response is a key phrase followed by a carriage return. Some responses can include explicit requests for changes in parameter values at the user's option. For those responses which can take up more than one line, a period terminates the response.

Some responses can contain a number of periods, and consist of more than one line, e. g. queries. Such responses are terminated by a key phrase, e.g. "End of query.". To eliminate problems caused by missing periods, etc., a user should be required to enter at least one character within 10 seconds of a carriage return; otherwise the multiple line response is considered complete. Such a rule is needed to prevent the system from waiting for user action while at the same time the user is expecting action by the computer.

Each reply text uses an ampersand "#" to indicate a mandatory carriage return. Additional carriage returns are inserted as needed by a console message director depending on the number of characters per line available on a

```
$ Proceed
- SMART.
- No title, minimal replies.
# Name?
- Mike Lesk
# Access Code
- XAQ13
#
#            SMART              XAQ13   Mike Lesk
# Relevancy decisions for active query 10?
- Yes.
# Document # 405  603  201  815  10004
- Decisions  3,4,3,5,1
# Abstract decisions?
- Yes.
# Relevance Feedback?
- Yes.
# Search?
- Yes, search.
#
#     Results of 3rd search of Query 10
.
.
.
- DONE
# Control is relinquished to the supervisor.
$ Proceed
```

Relevance Feedback

Fig. 2

specific console. A hyphen "-" indicates that the console keyboard is unlocked for a user response. Each quoted anticipated response, such as the key phrase responses, can be abbreviated by using only the capital letters specified in the response. All anticipated responses can be typed using any mixture of upper or lower case letters.

The contents of the 'Internal' column are, for the most part, self-explanatory. The use of the variable READY is described later but included in the Table for completeness. It indicates whether console interaction is needed, or whether internal work is needed.

The 'Next Segment' field indicates which segment is to be considered next. Often this is dependent on the response or the Action field. An "R" indicates a return to whichever segment was previously considered. Each user is assigned variables to indicate the segment he is in and the line of text (for that segment's message) that is being transmitted. When a console joins SMART, logical control is first set at segment 9 if SMART is on-line, otherwise control is set at segment 1. Note that segments above 104 are not included in the Table, but would be set up in the same way as other segments.

4. Console Driven Document Retrieval — An Internal View

This section describes a possible implementation of the on-line document retrieval system presented earlier. All routines available for batch SMART runs are usable without any reprogramming. An on-line executive program is however needed to drive the consoles and the batch routines.

A) The Internal Structure

The internal structure needed for a prototype system must satisfy several goals. As indicated in the introduction, a prototype system must

Segment Number	Reply Class	Messages for Consoles	Anticipated Responses from Consoles	Internal Action	Next Segment
0		(none)	"DONE"		51
			(Attention Key)	Delete transmission and activate keyboard	R
			"Tutorial Replies"	REPCLS = Tutorial	R
			"Short Replies"	REPCLS = Short	R
			"Minimal Replies"	REPCLS = Minimal	R
			"?" or an unanticipated response	If REPCLS = M Then REPCLS = S	R
				If REPCLS = S Then REPCLS = T	R
				If REPCLS = T	9000
1				SMART is on-line	2
				SMART is not on-line	3
2		#SMART is already on-line. You may not initiate a duplicate system.			51
3		#SMART is initiated. Your console is the master console. May other consoles attach to SMART?-	"Yes"	NEWCON = Yes	3.5
			"No"	NEWCON = No	3.5
3.5			(Reply Class Shift Only)		4

a) Introductory Segments

SMART Console Interface

Table 1

Segment Number	Reply Class	Message for Consoles	Anticipated Responses from Consoles	Internal Action	Next Segmen
4	S	#What is your name?-	User's Name	Store Name	6
	M	#Name?-			
6	S	#What is your access code?-	"None"	Assign an access code	7
	M	#Access code?-	Access code	Verify code-OK	8
				NOK	9900
7		Your access code is "ACCODE".		NUMCUS(-number of customers	
				ACCODE(-User's new access code	
				Store access code	100
8		#Welcome to SMART.			
		ACCODE		ACCODE(-access code	
		NAME		NAME(-User's name as on file	
				Does user have any unfinished queries?	
				Yes	2000
				No	100
9				If SMART is on-line	3.5
				If SMART is off-line	10
10		#SMART is not now on-line. Retrieval will be available (time, day).			51

a) Introductory Segments (contd.)

SMART Console Interface

Table 1 (continued)

Segment Number	Reply Class	Message for Consoles	Anticipated Responses from Consoles	Internal Action	Next Segment
50	S	#Please select one of the following programs... #Query, Analyze, Search, Display, Feedback, Pre-search, Search Options, Feedback Options, Analysis Options, Judgments, Done.	"Done."		51
			"Query."		100
			"Analyze."		500
			"Analyze using XYZ strategy."	ANALPV=XYZ	500
			"Search."		1000
			"Search, using XYZ strategy."	SEARPV=XYZ	1000
			"Display."		2000
			"Feedback."		3500
			"Feedback, using XYZ strategy	FEEDPV = XYZ	3500
			"Judgments."		3000
			"Pre-search."		4000
			"Analysis options."		5000
			"Search options."		6000
			"Feedback options."		7000
51		#Thank you for using SMART #Control is relinquished.		READY = 0 TST = 0 Return control of console to supervisor	

b) Central Director

SMART Console Interface

Table 1 (continued)

Segment Number	Reply Class	Message for Consoles	Anticipated Responses from Consoles	Internal Action	Next Segment
100		Do you wish to enter a query?	"Yes."		101
			"No."		50
101	S	#Please enter your MAXQUEth query. #Type "End of query." when finished.		MAXQUE = MAXQUE + 1 NUMQUE = MAXQUE	102
	M	#Enter MAXQUEth query.			
102		-	A line of a query.	Store line. Does line end in EOQ? YES	103
				No	102
103	S	#Is your query ready for analysis?-	"Delete Query."	MAXQUE = MAXQUE - 1 Delete query	101
	M	#Analyze?-	"Add to Query." "Boolean."	Does user want to supply Boolean Information? YES	104
				No	500
			"Yes, Search." "Yes."	DOANAL = 1 DOCENT = 1 DOSEAR = 1	500
			"Yes, Search, using XYZ Strategy."	As above and SEARPV = XYZ	500
			"Yes, using XYZ Strategy."	DOANAL = 1 ANALPV = XYZ	
			"No."		50

c) Query Text Handling

SMART Console Interface

Table 1 (continued)

have the speed and ease of use of a production system, as well as the flexibility and measurability of an experimental system. A document retrieval system must provide fast on-line service and exhaustive, inexpensive off-line service. A typical first thought is simply to provide two systems — one for on-line work, and the other for off-line work. However, a single, flexible system capable of handling both types of service is normally less expensive to develop, operate and maintain than two separate systems, provided a scheme with the needed features can be found.

The flexibility required to provide on-line and off-line service in a single package is best illustrated by the differing amounts of transmitted information. Off-line users will want, and can afford, to use large volumes of information. Such a volume of information cannot be transmitted at low cost to an on-line user, nor would an on-line user be able to cope with the quantity of information of use and interest to an off-line user.

Another illustration of the needed flexibility is related to machine storage. During off-hours, ownership of large amounts of storage for long lengths of time may be possible. Most on-line requests, however, will be serviced during the day when others also want to use the computer. To reduce costs, it is necessary that a minimum of computer resources be permanently allocated to each specific task. Unfortunately, human response times are much slower than normal computer response times when the computer is being used for batch processing. For example, a complete off-line search for 42 queries and 1400 documents can be completed in less real-time than a single on-line query because of the slowness of human response. (Obviously, the 42 query search requires more process time.)

B) General Characteristics of SMART Routines

To satisfy the need for flexibility and modifiability, SMART is programmed as a set of small, clearly defined, and well documented Fortran subroutines. Each subroutine accomplishes one task with a minimal interface with other routines. Each SMART routine lies in a distinct class depending on the amount of structure in the data used or manipulated. On the bottom of the pyramid are the I-O routines and the MOVE routines (which move sets of sequential locations from one place to another). These routines "know" only the length and origin of the fields with which they deal.

Next in the hierarchy are routines which deal with the various kind of vectors. SMART uses several kinds of vectors, all consisting of a "head" indicating the length of the vector followed by information in double words. In the case of concept vectors, these double words contain concepts and weights; in the case of result vectors, the first word contains the document number and rank retrieved (each in half words), and the correlation of the document with the query. The routines that deal with these vectors "know" the internal structure of the vectors. Some examples of this class of routine are LSTCON, which prints the contents of a concept vector, and RESULT, which prints the contents of a vector of document-query correlations.

Above this level are routines which deal with groups of vectors. These are the routines which know that many queries exist in the system. Typical of these routines is BLOCK, which combines the result vectors for the several iterations of one query during a batch run, and gives the combination, one query at a time, to RESULT.

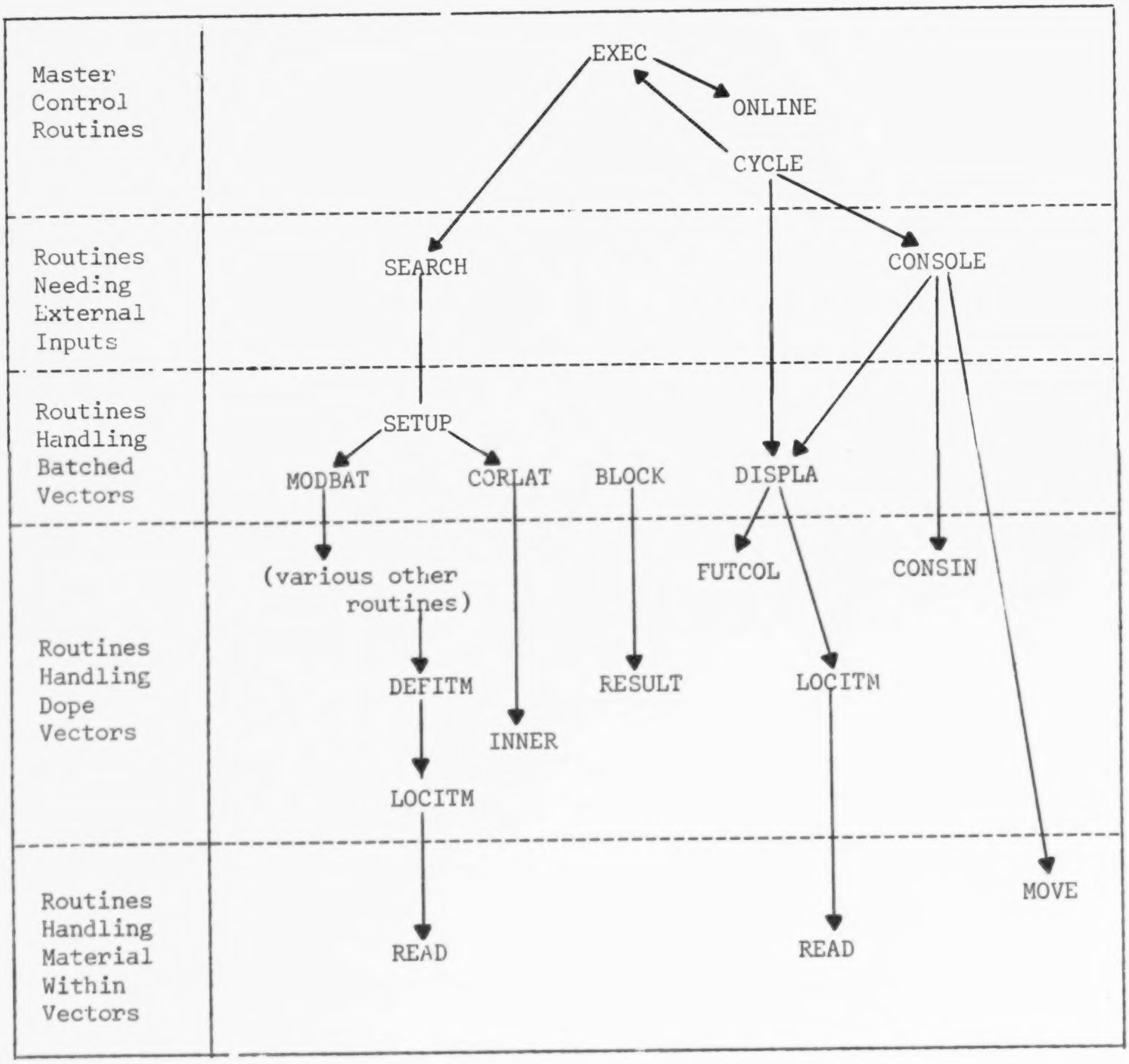
At the top to the entire pyramid are the routines EXEC and ONLINE. EXEC is a card-controlled driver for the system. It is normally used for batch experimental work and jobs typically done off-line, such as the addition of new text and centroid generation. ONLINE is normally used to control on-line document retrieval. A partial tree of SMART routines showing this structure follows in Fig. 3.

C) Pseudo-Batching

Basic to an understanding of the mechanism proposed for document retrieval is the idea of pseudo-batching. In any reasonable batch-processing document retrieval system, a large number of queries are handled in parallel. This serves to reduce the fixed overhead per query to a fraction of the total overhead. So long as the increased expense of dealing with several queries is kept small, there is a net gain in effectiveness per unit cost.

A basic problem in an on-line document retrieval system is that each search passes through different stages with different requirements. This presents problems because of the multiplicity of distinct programs which may be required, as well as the input-output problems. If each query is multi-programmed with other queries, severe competition for resources would result. One query would need document files, another dictionaries, and yet another would require text files. A complicated scheduling algorithm would be required to untangle the requirements for file access facilities and storage space; this would increase overhead costs sharply.

In an on-line system where many users individually cycle through the same set of routines and files, a much better utilization of resources results by batching the incoming queries. If the system processes only


Master Control Routines
Routines Needing External Inputs
Routines Handling Batched Vectors
Routines Handling Dope Vectors
Routines Handling Material Within Vectors
EXEC
ONLINE
CYCLE
SEARCH
CONSOLE
SETUP
MODBAT
CORLAT
BLOCK
DISPLA
(various other routines)
FUTCOL
CONSIN
DEFITM
RESULT
LOCITM
INNER
LOCITM
MOVE
READ
READ

Structure of SMART Routines

Fig. 3

those queries available at the start of a (twenty second) cycle, competition for resources is eliminated. Each query would then take thirty seconds on the average; twenty seconds of actual processing and ten seconds of waiting.

Many advantages can be accrued to the overall system and thus to the user by the batching of queries. Of greatest importance is the resulting lack of competition for different files or for space to store them. Secondly, each query has an apparent overhead considerably less than it would have if it were the only query to use a file at a given time. Obviously, lower overhead means lower cost.

D) Attaching Consoles to SMART

Since one can assume that consoles will not be continuously dedicated to a document retrieval system, at least in an experimental environment, provision must be made for transfer of control of a console from the computer supervisor to SMART. If SMART is core-resident and a specific console is wanted for SMART, the process is as simple as obtaining additional disk space or more core. However, it is desirable that a user be able to go to any available, supervisor-controlled console, and that the console be transferred to SMART at the user's initiation. Under such circumstances, the possibility also exists that SMART is not available on-line at some given time. Naturally, the problems and cost of serving additional users are far less when SMART is already on-line than when SMART must be started for the first user. Since SMART wishes to permit anyone to utilize the document retrieval system, provision must be made to prevent the occurrence of unreasonable expenses. One obviously unreasonable expense is the improper activation of SMART. Another problem is the need to keep to a minimum the actions which the typical, non-computer-oriented user must carry out to use the SMART system on-line.

For these reasons SMART could include a small routine that is continuously a part of the supervisor. Normally, after a user has activated a console (e.g. by dialing the computer if telephone lines are used), the computer expects the name and account number of the user (in order to prevent unauthorized usage). The user may then enter simply the word "SMART". This will cause the execution of a program called SMTLATCH which is supplied with the "name" of the console presently wanting SMART. This code will "know" whether SMART is on-line or not.

If SMART is not on-line an appropriate response is made. (An example is presented in Fig. 4.) If SMART is on-line, the console number of the new user will be made available to the normal SMART programs and a flag will be set indicating that a new console needs to be attached. When SMART regains use of the computer, the supervisor can be requested to transfer control of that console to SMART.

```
(Dial computer and press carriage return.)
#Proceed.
%SMART.
$SMART will be available next at 3 p.m.
   Tuesday, October 4, 1968.
#Proceed.
%
```

Console Response to a Request for SMART
When SMART is not On-line

Fig. 4

E) Console Handling — The Supervisor Interface

SMART will not need to worry about physical control of the consoles. Rather SMART provides a routine which the supervisor can call whenever a new line is available from a console. The console keyboard is then locked (i.e. nothing more can be typed by the user) until SMART allocates space for a new line somewhere in a SMART section of memory and so tells the supervisor. Alternatively, at this time, SMART can transmit a line to the console. Normally the console keyboard will be freed fast enough (if multi-line input is anticipated) so that the user will be unaware of the keyboard ever being locked.

When SMART wishes to write on a console (which includes unlocking the console keyboard), a call to the supervisor is made with the location of a message and the name of specific console on which the message is to appear. If the keyboard of that console is locked, the message is immediately transmitted. If the keyboard is not locked, the transmission is refused and SMART will have to lock the keyboard first and accept whatever message was transmitted. (On the equipment presently available the console cannot be locked; only the user can lock the keyboard by pressing "Attention" or "Carriage Return"; the system must therefore wait for user action.)

F) Parameter Vectors

As each enquirer is introduced to SMART, he is associated with a user vector that contains pointers to parameter vectors. These vectors are filled with information taken from control cards during a batch processing run, or from a default vector for new on-line users, or from personal parameter vectors. These parameters supply values needed to control the action of the retrieval routines. Each user may define his own personal parameter vectors which can be saved for use on many searches.

G) The Flow of Control

The flow of batched queries is comparatively simple compared to that of on-line queries. Although batched and on-line queries use different means to fill parameter vectors, and take different action with respect to the output of most routines, these differences are unimportant.

The manner of introducing an on-line user has already been described. (As far as SMART is concerned, a user and the console he is then using are equivalent in all ways. Thus, wherever the word 'console' appears, the word 'user' could be substituted.)

The on-line control program consists of two logically distinct routines. CONSOL handles physical communications with the consoles on an interrupt basis (i.e. in real-time). CYCLE handles the use of core and the large system files by cycling among them, satisfying users as it can. Logical control of each console shifts between CONSOL and CYCLE.

The SMART On-line Console Control Block (SOCCB) indicates at any given instant which routine is logically in command of a console. The SOCCB synchronizes the real-time routine CONSOL with the process-time routine CYCLE. The READY flag associated with each console takes on certain values if the console is awaiting completion of a task done by CYCLE. When CYCLE is finished, the READY key is changed. Since the key is changed, CONSOL can recognize that it should proceed with that console.

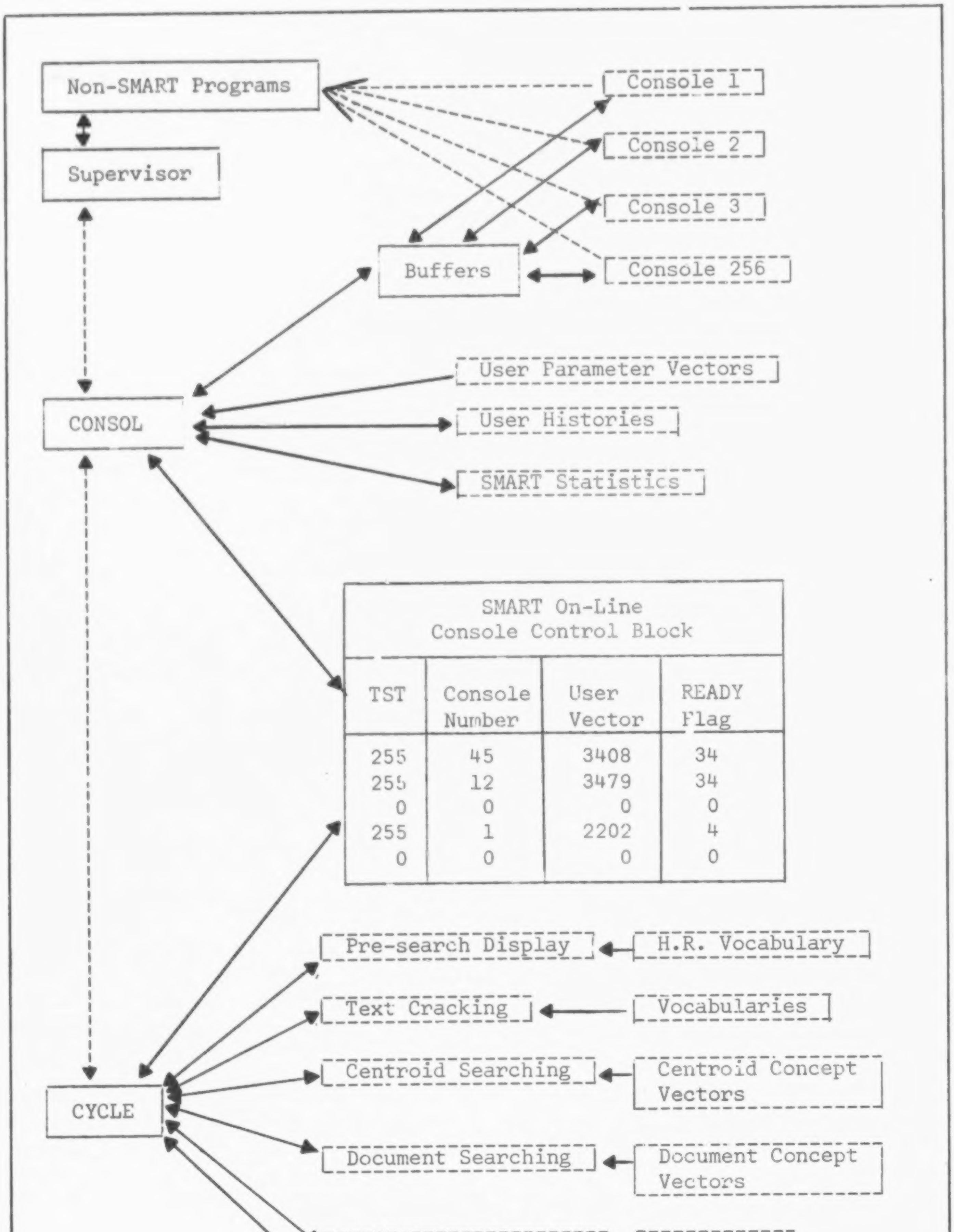
Testing READY flags (for up to 256 consoles) is accomplished by a single instruction (Translate and TEST — TRT) using a 256 byte array. Since the test is fast, it can be carried out frequently by both CONSOL and CYCLE. For example, after sending each line of a message to one console, CONSOL can test to see if any other console requires service for a single line. If so, the servicing of the one console with a series of lines is terminated, and consoles with single-line needs are handled. CONSOL then returns to the multi-line message and finishes. CYCLE uses the speed of the TRT instruction to locate those queries needing a specific process. After each CYCLE driven routine finishes with a batch of queries, the table can be scanned to see if, in the meantime, any other queries now need that same process. Some routines which can be logically divided into two parts, one essentially in-core and the other necessitating file accessing, could be programmed to check for "latecomers" to speed up overall response without losing the advantages of cycling.

For a list of typical READY flags see Table 2.

H) Timing Considerations

In order for the type of organization presented to be acceptable to non-SMART users of the computer, two timing considerations are paramount. First the CONSOL routine must be assigned highest priority by the supervisor, since it must respond to on-line signals. CYCLE is assigned the second highest priority. This implies that if CYCLE is free to perform work, the CPU is taken away from any other executing program (except CONSOL and the supervisor itself). Normally, however, CYCLE is I-O bound. While CYCLE is waiting for needed information from noncore resident files, and when CYCLE has no work to do, the CPU is able to do the work of other customers.

Thus, CONSOL must have available everything it needs to work and CYCLE must contain no wait loops of any size. If information is not available, the supervisor must be given control until the required information is available.


Non-SMART Programs
Supervisor
Console 1
Console 2
Console 3
Console 256
Buffers
CONSOL
User Parameter Vectors
User Histories
SMART Statistics
SMART On-Line
Console Control Block
CYCLE
Pre-search Display
H.R. Vocabulary
Text Cracking
Vocabularies
Centroid Searching
Centroid Concept Vectors
Document Searching
Document Concept Vectors
Post-search Display
H.R. Text
Query Update
Document Concept Vectors

SMART On-Line Console Control Block

TST	Console Number	User Vector	READY Flag
255	45	3408	34
255	12	3479	34
0	0	0	0
255	1	2202	4
0	0	0	0

Legend: Core-resident ———— Auxiliary ----- H.R. Human Readable

SMART On-line Control Logic

Fig. 5

Routine Needed	READY	Meaning
(NONE)	0	Unused slot.
CONSOL	1	Newly arrived console, no assigned user vector.
(NONE)	2	Console keyboard unlocked for user transmission.
CONSOL	3	Console keyboard locked by receipt of a user transmission.
(NONE)	4	One line message going to console.
CONSOL	5	Console keyboard locked further lines are needed.
CYCLE	6	Allocate core.
CONSOL	7	Core Allocated.
.	.	.
.	.	.
.	.	.
CYCLE	20	Crack text.
CYCLE	21	Cracking text.
CONSOL	22	Text cracked.
CONSOL	23	Notifying user.
CYCLE	24	Set-up pre-search display.
CYCLE	25	Setting-up pre-search display.
CONSOL	26	Pre-search display setup.
CONSOL	27	Displaying to user.
CYCLE	40	Search centroid tree.
CYCLE	41	Searching centroid tree.
CONSOL	42	Centroid tree searched.
CONSOL	43	Informing user of results of tree search.
.	.	.
.	.	.
.	.	.

READY Flags

Table 2

I) Noncore Resident Files

Before going into CONSOL and CYCLE in detail each of the files used by SMART is introduced briefly. The various logical segments of core are then similarly defined to provide a reference and to eliminate detailed descriptions within succeeding sections.

SMART files can be divided into three distinct classes — those used by CYCLE, those used by CONSOL, and the consoles themselves. The console files are basically standard sequential files, with, however, an unpredictable access time. Like sequential files, records are read (or written) one-at-a-time and in linear order. There is, of course, no backspacing, rereading or overwriting.

CONSOL deals with three files of a more familiar nature. The 'SMART Statistics File' is a sequential, write-only file on which is placed information to enable evaluation of SMART's performance by supervisory staff. Information such as observed user and SMART response times, and statistics on query authors using the system might be kept.

The 'User History File' retains information about unfinished queries on an individual user basis. For each user, such information as the number of queries he has submitted to the system, the number still active, and accounting information may be kept. For each active query, a record is kept of the text of the original query, and of the last active concept vector for that query. Perhaps, a list of additional documents, unseen by the user, should be kept to try to forestall a complete lack of positive feedback. In this manner costs could be kept reasonably low for a majority of users by not showing many documents except when necessary. One might also want to keep some type of record of the searched centroid tree so that "obviously" unsuitable tree nodes would not have to be reconsidered during relevance feedback.

The 'User Parameter Vector File' contains user parameter vectors. Each user can have several different parameter vectors (with distinct names) for different purposes. The only reason for separating this file from the previous file is that this file is essentially a read-only file, whereas the previous file is updated with every system access. It is anticipated that the directory for this file would be one part of the preceding file.

The files used by CYCLE-called routines are of two distinct types — human readable and machine readable. The human readable files contain information suitable for display to normal users at consoles. The other files are however organized for maximum speed of access and minimum space for storage of information used solely by SMART. A complete system must have human readable files — the vocabulary aid files and the source text files. Vocabulary aid files contain thesaurus expansions, hierarchies, frequency lists, etc. Source texts contain titles and abstracts of documents in a form suitable for on-line display. Normally vocabulary aids are used prior to a search and texts after a search.

There are three machine-readable classes of files — vocabulary files, files of centroid concept vectors, and files of document concept vectors. Vocabulary files contain the information needed to quickly understand input text (i.e. to convert raw text into a standard concept vector). The other two files contain, respectively, files of centroid concept vectors and files of document concept vectors. The separation of centroid and document concept vectors into two distinct files is dictated by the relative sizes of the two files. Commonly, a centroid has over 10 sons: thus a centroid tree for a file of 100,000 document would contain less than

9,000 nodes. In most situations, the centroids could be accessed faster as a separate data set because of their smaller volume.

There also exists a file which contains the programs called by CYCLE. In order to further conserve space, it may be desirable that these mutually exclusive routines be overlayed during execution.

J) Core Resident Files

Seven types of core resident files are used by SMART. They have differing typical lifetimes, lengths, sources, and destinations. Because of their differing lifetimes, they are allocated from different pools of available core. This minimizes a serious tendency to fragment core and eliminates a need for dynamic relocation of in-core files. By permitting the system to obtain variable amounts of core, SMART is able to work in 50 K or 500 K, albeit with grossly different response times and CPU utilization rates.

The first file is the previously mentioned SMART On-line Console Control Block (SOCCB). This block is the key to the entire control of the on-line system and is, therefore, described in detail in the next section. The size of the SOCCB is fixed when SMART is initiated by the number of consoles to be accepted on-line at one time. This block is retained until SMART goes off-line.

Each user is assigned a user vector. This block is of fixed length and is retained as long as the user is on-line. The user vector contains pointers to the locations of dynamic fields "owned" by the given console. These fields include parameter vectors, buffers and correlation vectors. The user vector is accessed only by CONSOL and CYCLE.

The parameter vectors contain values for variables used to control the various routines. Each routine needs its own parameter vector. There exists a standard default parameter for every routine, and these standard vectors are core-resident for the life of a given invocation of SMART. Any user vector can point to one of these default vectors; however, no user can change values in the default vectors. If a user wishes to change any values, space is allocated for his own individual parameter vector for each routine the user wishes to control in a non-standard fashion. A user may name his parameter vectors in order to re-use them easily. An individual parameter vector is core-resident only for the duration of the process which that vector controls.

Buffers contain a line or a track of information. They typically have a short lifetime, and the space occupied by the buffers is reutilized at a high rate. Buffers to or from a single file can be linked while in-core. These vectors constitute the majority of core needed by SMART. In some cases, it may be desirable to keep some buffers in core in anticipation of repeated use. If sufficient core is available, this can be done. However, this in-core saving of a buffer is unknown to all routines except to the buffer manager. This permits a routine to use 50 K of 500 K bytes without any internal knowledge. Only the response times to requests for a buffer will differ depending on the amount of core utilized.

The concept vectors constitute the output of the routines converting text into concept vectors, and of the query update routines. These vectors are much shorter than the text they represent, and they can be more easily utilized for search purposes. Only one concept vector per user need be kept in core and the concept vector supplants the buffers containing the original query.

Specification and correlation vectors contain the names of individual centroids or of documents to be matched with a query, and later, the correlations with those items. The life of these vectors is short but the core requirements for a single query can be determined only dynamically.

Result vectors are shortened correlation vectors. They are used by CONSOL to pass information to the consoles.

5. CONSOL — A Detailed Look

Once the overall structure of the proposed on-line system is understood and the contents of the various files in understood, a detailed explanation of the operation of the two major routines becomes straightforward.

CONSOL will be considered first since it is first logically. Before going into the routine itself, the SMART on-line console control block (SOCCB) is described:

A) Competition for Core

It is possible that one user may finish a line and the interrupt-called supervisor can start CONSOL, while a second user can finish his line before CONSOL finishes with the first user. The second user's finish would cause the supervisor to start CONSOL again. A routine like CONSOL is called reentrant if several different processes (users) can simultaneously execute it. On a single CPU machine like Cornell's 360/65 the simultaneity is apparent and due to interrupts. However, on a multiple CPU machine the sumultaneity could be real. In both cases the problem is the same: no process can know if another process is also executing the same code. The requirement is that no "edition" of a reentrant routine can change core locations possibly known to another "edition" of that routine. If the reentrant routine must obtain additional core, the same problem exists — two editions may try to take the same space. A similar problem arises between CONSOL and CYCLE: CYCLE could be claiming an area of core while at the same time CONSOL decides to use that same area.

In order to prevent destructive competition for ownership of resources, the 360 provides a single instruction which locks a resource as it tests that resource for availability. The instruction is called Test and Set (TST). Basically TST sets a byte non-zero and sets the condition code to zero or non-zero as the previous contents of the byte were zero or non-zero in one inseparable step. (The TST instruction is outlined in Fig. 6).

B) The SMART On-line Console Control Block

The SMART On-line Console Control Block (SOCCB) shown in Fig. 5 holds four items for each active user. The maximum number of consoles that can be on-line at one time is decided when SMART is first entered; MAXUSERS contains this number. The fields marked TST and READY (in Fig. 5) are each vectors of "MAXUSERS" consecutive bytes. The TST field contains zero if that particular line is unused. When a line is reserved for a particular console, the TST field is set non-zero. The Console Number field contains the supervisor number for a console and the User Vector field contains the location of the user vector for that console.

TST LOCK (Instruction)		Before Execution	After Execution
Case 1	Location LOCK	0	255
	Condition Code	-	zero
Case 2	Location LOCK	255	255
	Condition Code	-	non-zero

The Test and Set Instruction (TST)
as Applied to the Location Named LOCK

Fig. 6

C) The READY Flag and the TRT Instruction

The READY field contains one of 256 equivalent flags. Each flag (value) indicates what process is then needed by that user. Typical values are given in Table 2. To understand the value of the vector, one needs to understand the Translate and Test (TRT) instruction. This instruction considers two read-only vectors. The first vector is the vector of READY values; the second contains a table of 256 bytes. This last table contains zero bytes except in those bytes whose address (relative to the first byte of the table) is the same as a READY value which must be tested. The TRT instruction takes bytes from the first vector, one-at-a-time, and looks at the table entry corresponding to the value of that byte. If the object byte is zero, the next READY value is considered; if the object byte is non-zero, the instruction ceases with that object byte and the location of the source byte is made available. If no byte stopped the instruction, that fact is so indicated. If the instruction is stopped by a non-zero object byte, the registers used by the instruction are left in a condition such that the instruction can be reexecuted for the remaining bytes in the source vector. A pictorial explanation of the TRT instruction is given in Appendix 1.

For internal convenience, READY values are often assigned in blocks — each block associated with a given process. Most processes can be divided into four phases: unconsidered by CYCLE, being considered by CYCLE, unconsidered by CONSOL, and being considered by CONSOL. Some READY values appearing in Table 2 show this assignment.

D) The Routines LATCH, CONSIN, and CONSOT

When a person types "SMART" on a console, the supervisor transfers control to SMTLATCH. SMTLATCH interrogates the variable SMTOPEN. If SMTOPEN is zero, SMTMSG (containing the appropriate message) is sent out to the calling console. If SMTOPEN is non-zero, control is transferred to (the location contained in) SMTOPEN. SMTLATCH, including SMTOPEN and SMTMSG, is always available to the supervisor as a standard supervisor process. Since SMTLATCH takes only 96 bytes, it can be kept constantly core-resident.

The first routine called when SMART is started in the standard manner is (ONLINE) which inserts the location of the routine LATCH at SMTOPEN. When SMART no longer wishes to accommodate new users, the routine OFFLINE updates SMTMSG to indicate the next scheduled time for on-line document retrieval; finally, SMTOPEN is set to zero. Consoles active in the system can still be accommodated in any suitable manner.

When LATCH is called, an unused row is located in the SMART On-line Console Control Block (SOCCB) using the TST to insure that the selected row is indeed available. LATCH then changes READY for that row to 1 (from 0) and stores the name of the console in the SOCCB. If CONSOL is running, LATCH simply returns to the supervisor (which will restart CONSOL where CONSOL was interrupted). If CONSOL is not running, LATCH causes the supervisor to mark CONSOL as runnable. LATCH then returns to the supervisor. The new console will be noted in due course by a TRT in CONSOL.

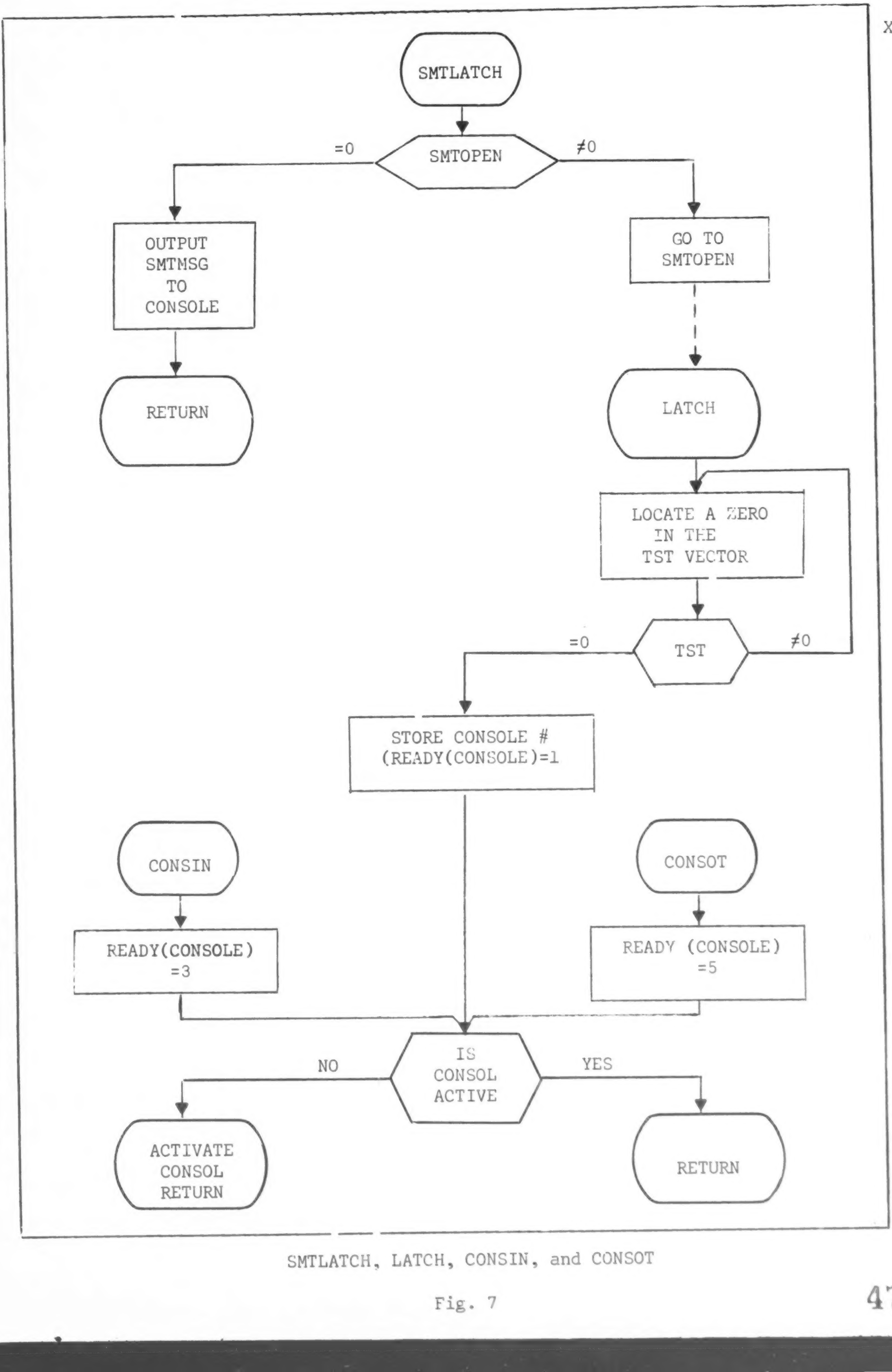
Routine CONSIN is similar to LATCH; when a console is released to a user, the supervisor needs the name of a routine to call when the transmission from the user is complete as well as a place to put the transmission. CONSIN is that routine. The supervisor tells CONSIN the name of the console which interrupted; CONSIN then changes the READY flag for the console (from 2) to 3 and insures that CONSOL is running.

To minimize over-all response times only one line will be set up for transmission to a console if another console also needs service. If a console needs several lines, but only one is transmitted, CONSOL will have to prepare other lines at a later time. To do this on an interrupt basis, routine CONSOT is called by the supervisor after transmission of a line to a console if that console requires more information.

All of these routines consist of fewer than a hundred instructions and take less than a millisecond to execute. Fast response to the changes made in the READY table is insured, since CONSOL tests the flags after each line of a transmission is complete. The test for a console needing attention is less than fifteen microseconds if no console needs attention (assuming ten on-line consoles). Since the test is so fast, frequent repetition is not expensive.

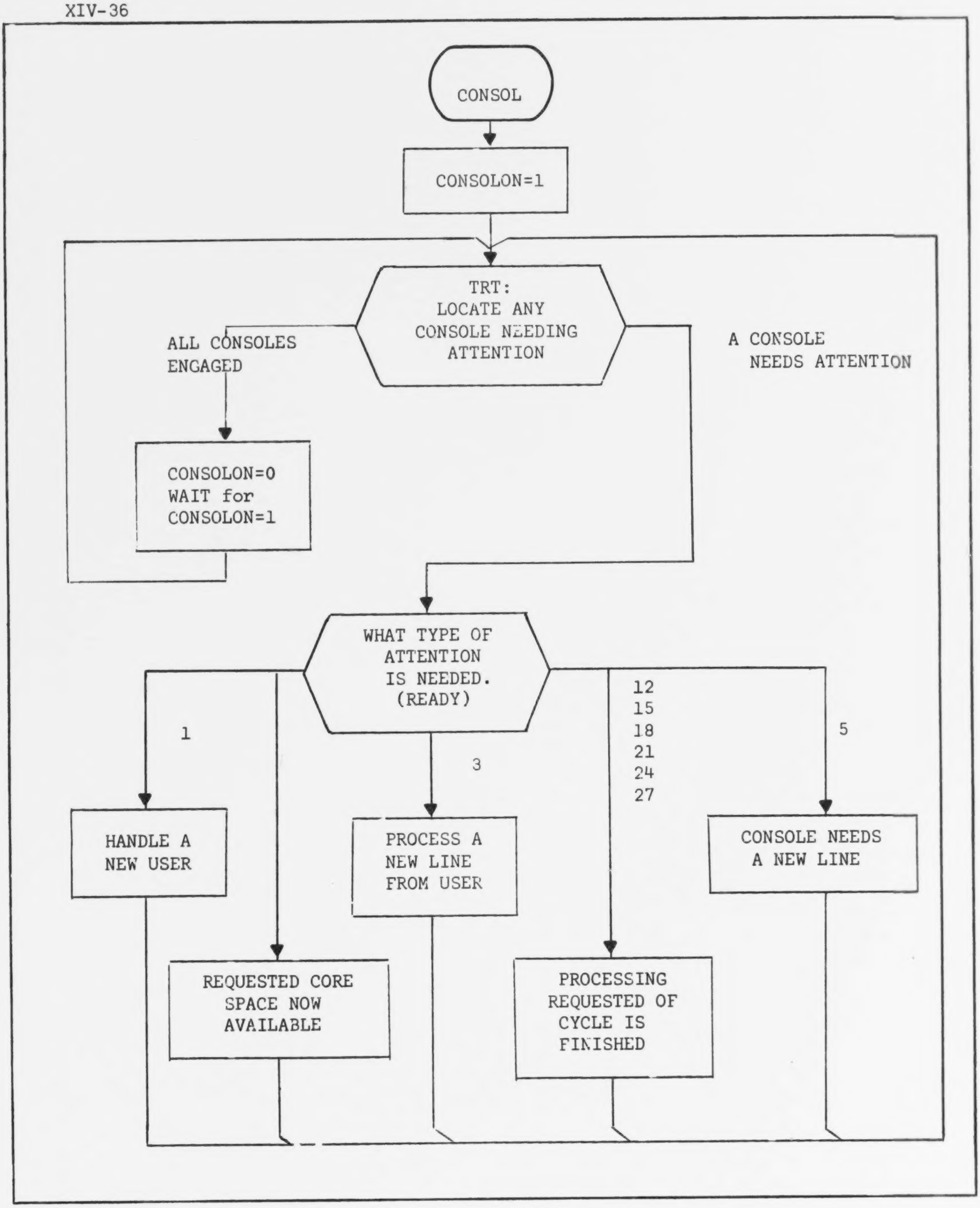
E) CONSOL as a Traffic Controller

In basic terms, CONSOL uses the TRT instruction to select a console which has a need and then satisfies the needs of that console at least temporarily. CONSOL then uses the TRT again to select another console. Eventually all console needs will be satisfied and CONSOL will retire to permit other processes to use the CPU; one of these processes will most likely be CYCLE. When CYCLE has completed a request for a user, or a set of requests, CYCLE will ask the supervisor to restart CONSOL, and, by so doing, suspend itself in real-time (but not in process-time). Alternatively, the completion of a user line at a console will result in an interrupt-initiated call to CONSIN or LATCH which can wake-up CONSOL. Effectively then, CONSOL uses the TRT instruction to facilitate a traffic direction problem.


SMTLATCH
SMTOPEN
=0
≠0
OUTPUT SMTMSG TO CONSOLE
RETURN
GO TO SMTOPEN
LATCH
LOCATE A ZERO IN THE TST VECTOR
TST
=0
≠0
STORE CONSOLE # (READY(CONSOLE)=1
CONSIN
READY(CONSOLE) =3
CONSOT
READY (CONSOLE) =5
IS CONSOL ACTIVE
NO
YES
ACTIVATE CONSOL RETURN
RETURN

SMTLATCH, LATCH, CONSIN, and CONSOT

Fig. 7


CONSOL
CONSOLON=1
TRT:
LOCATE ANY
CONSOLE NEEDING
ATTENTION
ALL CONSOLES
ENGAGED
A CONSOLE
NEEDS ATTENTION
CONSOLON=0
WAIT for
CONSOLON=1
WHAT TYPE OF
ATTENTION
IS NEEDED.
(READY)
1
3
12
15
18
21
24
27
5
HANDLE A
NEW USER
PROCESS A
NEW LINE
FROM USER
CONSOLE NEEDS
A NEW LINE
REQUESTED CORE
SPACE NOW
AVAILABLE
PROCESSING
REQUESTED OF
CYCLE IS
FINISHED

CONSOL

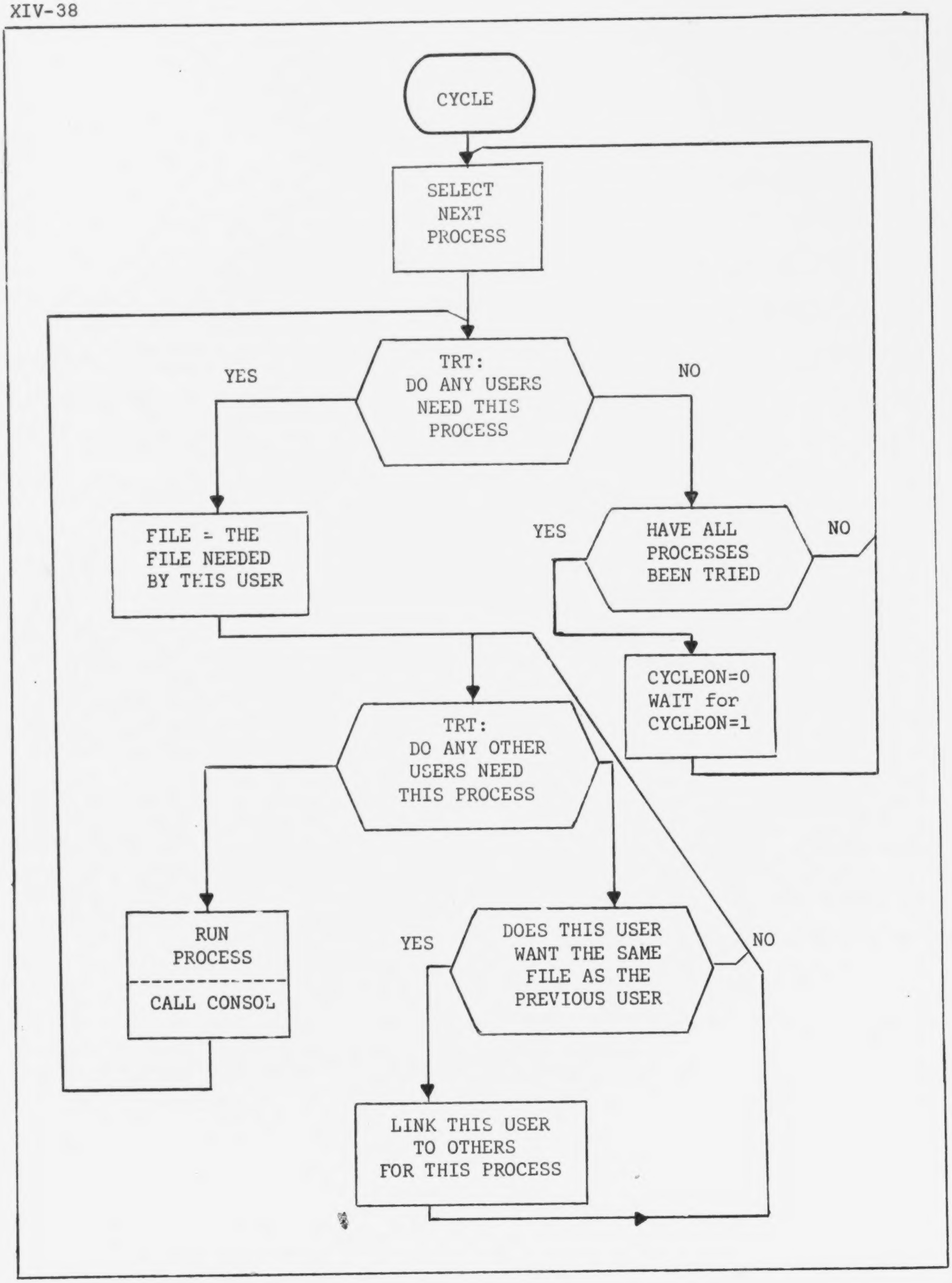
Fig. 8

It is apparent from a scan of various possible needs that some are more urgent than others. For CONSOL, however, needs are satisfied so quickly that the arbitrary selection of the console highest in the SOCCB is adequate. CONSOL works so fast that even if the 256 users were on-line and all had a need at the same instant, and the first user were serviced first, the last user would be satisfied before the transmission to the first user was complete. In actuality, in most cases, only one user will need service at any given time. The obvious exception to this is after CYCLE completes a task — at that time, several consoles will need transmissions. It is immaterial, however, which console is satisfied first, since all consoles will be satisfied by CONSOL in much less time that was taken by CYCLE.

F) A Detailed View of CYCLE

In contrast to CONSOL, CYCLE follows a strict pattern in deciding what to do. Like CONSOL, CYCLE uses the TRT instruction but CYCLE decides what process to do first. Then it sees which consoles need that process. If no console needs that process, CYCLE tries the next process in its list of processes. To permit on-line access to more than one collection for test purposes, or access by sophisticated users with special needs, each process is run for all consoles that request one collection and then for all consoles that require another collection. This is illustrated in Fig. 9.

Some object processes started by CYCLE are standard programs used for batch experimentation; text cracking, centroid tree searching, document correlation, and query redefinition. The processes unique to the on-line system divide into two classes — those that access files for the user and those that service CONSOL. There are presently two programs of


CYCLE
SELECT NEXT PROCESS
TRT: DO ANY USERS NEED THIS PROCESS
YES
NO
FILE = THE FILE NEEDED BY THIS USER
HAVE ALL PROCESSES BEEN TRIED
YES
NO
CYCLEON=0 WAIT for CYCLEON=1
TRT: DO ANY OTHER USERS NEED THIS PROCESS
RUN PROCESS
CALL CONSOL
DOES THIS USER WANT THE SAME FILE AS THE PREVIOUS USER
YES
NO
LINK THIS USER TO OTHERS FOR THIS PROCESS

CYCLE

Fig. 9

the first type: to display pre-search information, e.g. thesaurus categories, and to display post-search material, e.g. abstracts. Since CONSOL operates on an interrupt basis, it cannot allocate resources for itself. However, CONSOL does need to be able to obtain core storage space on demand. To provide this, CYCLE can be asked to allocate storage for a console and return control to CYCLE.

From the flowchart for CYCLE shown in Fig. 9, it can be seen that CYCLE restarts CONSOL without testing if CONSOL is running. This is possible since CYCLE can use the CPU only when CONSOL is inactive.

6. Summary

On-line information retrieval is implemented by two co-routines, CONSOL and CYCLE. The former operates in the real-time of the console user providing rapid response; the latter in the process-time inherent in any routine which needs to access auxiliary storage providing realistic costs for work done. The two routines communicate through a single area of mutually known core.

This system should prove adequate for both experimentation and real-time use in a library, for both the novice user and the sophisticated researcher with the complex problem.

Appendix

TRT READY, TABLEn (Instruction image, n=1, 2, 3 or 4)

Location: Contents:	READY + 0 1 2 3 4 5 6 7 8 5 4 3 4 2 0 0 0 1
Location: Contents:	TABLE1 + 0 1 2 3 4 5 6 0 0 0 6 0 0 0
Location: Contents:	TABLE2 + 0 1 2 3 4 5 6 0 0 0 0 9 0 0
Location: Contents:	TABLE3 + 0 1 2 3 4 5 6 0 2 3 0 0 0 0
Location: Contents:	TABLE4 + 0 1 2 3 4 5 6 0 8 0 0 0 0 0

Execution:	1st			2nd			3rd
Register:	0	1	cc	0	1	cc	cc
n							
1	6	(READY)+2	1			0	
2	9	(READY)+1	1	9	(READY)+3	1	0
3	3	(READY)+4	1	2	(READY)+6	1	0
4	8	(READY)+6	2				

(READY) means the address of READY ; cc = condition code

The Effects of the Translate and Test Instruction (TRT) When the Vector READY is Entered Against Several Tables

XV. Template Analysis in a Conversational System

S. F. Weiss

Abstract

This study presents a discussion of natural language conversational systems. The use of natural language rather than fixed format input in such a system makes possible the implementation of a natural dialogue system, and renders the system available to a wide range of users. A set of goals for such a system is presented. These include the provision of fast responses, usable by all levels of users, and the use of intellectual aids such as tutorials.

An experimental conversational system which meets these goals is implemented using a template analysis process. Template analysis is used not only to analyze natural language input, but also to control the overall operation of the process. Experiments with a number of users show that the system is easy to utilize and provides accurate analyses. A detailed discussion of both user and system performance is presented.

1. Motivation

Programs and data are normally entered into a computer in a batch processing mode. However, the recent trend in computer system design has been toward the development of large time shared systems which give a number of users simultaneous on-line access to the computer. This makes possible the implementation of conversational programs which permit real-time man-machine dialogues. Such conversational programs are both useful and necessary to cope with the ever expanding complexity of computerized data processing tasks. Consider for example, an on-line programming language such as APL. The ability to test amd debug a program on-line is an aid to the programmer. Errors are more easily located and may be corrected immediately. In addition, on-line data entry allows the programmer to adjust parameters and data while the program is running in order to get the desired results.

Conversational programs are also useful in all forms of language processing and expecially in information retrieval. Consider for example a case in which a natural language analysis program encounters an unresolvable ambiguity. In the batch mode, the program would be forced either to give up or to proceed using the multiple interpretations. But in a conversational mode, the system can ask the user for clarification and then proceed with perfect information as is shown in the example in Fig. 1.

U: TYPE 2 GRAMMARS

S: YOU HAVE USED TYPE AMBIGUOUSLY. PLEASE SPECIFY:

A. PRINTING

B. VARIETY

U: B

S: PROCEED

User Disambiguation

Fig. 1

In information retrieval the applicability of conversational programs is very broad. It is the only way to make the retrieval operation fast enough for practical use. In addition it permits the user to see results immediately and adjust his query and other search parameters to tailor the performance to his exact needs. The conversational mode is also the best framework in which to implement the relevance feedback process [11,24]. In general the conversational facility is an extremely powerful information retrieval tool.

Section 2 of this study discusses some existing on-line systems. Most of them require a fixed format input. But the current trend in information processing is toward natural language input. Not only does this permit the treatment of documents and queries in their original form, but it also makes the on-line facility available to a broad spectrum of potential users. This is especially important since on-line systems permit remote access from places such as libraries and schools which are not inhabited strictly by computer people. This study discusses conversational systems in general and presents a natural language facility for information retrieval.

There are four basic goals which any such natural language conversational system should meet. First, the system obviously must accept natural language input. Second, it must provide fast response. Users tend to become impatient if the delay between the submission of a command and the system's response exceeds more than a few seconds. Third, the system should be usable by all levels of users. Inexperienced users should be able to perform useful work. At the same time the system must not hamper the expert with excessive verbosity and unwanted material. And finally, the system should provide some intellectual aids such as tutorials and prompts which can help the user conduct a useful dialogue.

2. Some Existing Conversational Systems

Many conversational systems are currently in operation. Most are part of a larger implementation such as an information retrieval system. But a few such as ELIZA are designed solely to perform conversation. The major differences among the conversational aspects of the various systems is in the amount of man-machine interaction permitted. In some systems the on-line input is not far removed from batch input and the user has little control over the running of the process. At the other extreme are systems in which the user is directly linked to the process and is continuously in command of program operation. The discussion of on-line systems presented below is roughly in order of increasing complexity of dialogue.

The most basic type of conversation consists of a simple user input which results in some appropriate system action being performed. RECON [16], DIALOG II [29], TIP [15], and AUTONOTE [22] are representative of this type of conversation. In RECON for example, the user presses a button which indicates the desired operation and then types the operands on the console. In the other systems the user types the operator name followed by operands. Thus all these processes require a fixed input format. In addition, should the user become lost or confused, the systems cannot supply any intellectual aid to help him out. One type of user aid, the tutorial, is a feature of the AUDACIOUS system [2]. In addition to the normal operator-operand commands like those above, AUDACIOUS permits two special commands: HELP and PUNT. In response to these, the system produces a tutorial message appropriate to the user's position in the dialogue. In this way the confused user can receive help.

A second type of intellectual aid is the prompt. SPIRES [21] is an example of a system which uses the prompting feature. Unlike tutorials, prompts are presented without user request. Their purpose is to indicate to the user what type of information is to be specified in the current input. However, since prompts are presented without a user request, they can sometimes be a nuisance to the expert user. All the conversational systems presented thus far share two attributes. First, they all require fixed form input. And second, they are all information retrieval systems and hence the conversational operation was not the prime consideration in their development. The systems discussed in the next few paragraphs are designed basically to conduct conversation in natural language.

Probably the most famous natural language conversational system in Weizenbaum's ELIZA [34]. The program conducts a coherent dialogue with the user much like that between a psychotherapist and his patient. Inputs are searched for the presence of certain keywords and structures. These indicate the type of output appropriate to the input. For each input form there is more than one allowable response. ELIZA cycles through this set thus eliminating repetition and producing a more realistic looking conversation. The approach to conversation used in the system presented later in this study is similar to the ELIZA concept.

Another area of usefulness for conversational capabilities is in computer assisted instruction. One such conversational CAI system is Bolt's Socratic Instruction [6]. Its operation is basically an extension of the techniques outlined for ELIZA. Like ELIZA, the Socratic Instructor uses the user position in the dialogue along with the input to determine the proper response. In addition, the Socratic Instructor remembers all previous user inputs and dialogue points. These are also used in output determination.

Most conversational systems in existence today are implemented by basically ad hoc programming methods. This is not unusual for a fairly new area such as conversational programs. However, as on-line systems become more common, higher level implementation processes must be developed. One such process already in existence is the LYRIC system developed by Silvern [26]. This is a programming language for describing conversational CAI programs. With processes such as this, the conversational implementer is relieved of some of the ugly programming details in much the same way as a compiler-compiler aids the systems programmer.

The conversational systems presented here by no means constitute the complete set. They are, however, representative of most systems. It appears that systems such as TIP and SPIRES which perform efficient on-line information retrieval require highly structured input format. On the other hand those such as ELIZA which permit natural language input have a very weak concept of understanding. It would be desirable to develop a system which combines the best attributes of both; that is, a fast and accurate information system which allows natural language input. This is the topic of the following sections.

3. Goals for a Proposed Conversational System

This section discusses the design considerations that go into the development of a new conversational information retrieval system. Some elements of the new system are drawn from existing facilities while others are new. The primary goal of this system is to allow a user to conduct a natural language dialogue with the system. The only limitation is that the input be restricted to an information retrieval context. Not only should the user be allowed to specify natural language commands, but also there should be no restriction on the number of commands per line as there are in most other conversational systems. An input such as

> USE THE COSINE CORRELATION ON THE CRANFIELD COLLECTION.

should be perfectly legal. Of course there may be some inputs for which natural language is impossible or impractical and a fixed format input must be used. For example, the user should be required to specify a fixed form "SIGNOFF" in order to prevent accidental termination of the conversation. But these formatted inputs should be kept to a minimum. Another goal for this system is to be able to resolve automatically ambiguities occurring in the user's input. In addition the system must meet the requirements specified in section 1. These include providing fast response, being usable by all levels of users, and providing intellectual aids such as tutorials and prompting.

This proposal makes demands on the user as well as the system. First, the basis for learning the system is a manual. It would be aesthetically pleasing to allow the system itself to contain a computer aided instruction facility (CAI) which would make the system completely self-contained. Unfortunately this is impractical. Successful CAI requires concentrated and frequent exposure to the teaching medium. It appears that the typical information retrieval user dialogue will be both brief and fairly infrequent. Also, trying to teach the user at the console unnecessarily ties up the facilities. Thus an off-line approach to learning the system seems more reasonable. While no CAI facility is provided, the system should offer a prompting option by which a user can be led step by step, through a simple retrieval process. In this way the user may learn something about the system while actually performing useful retrieval work. The user's manual for this system is divided into several sections. Each deals with system use in progressively greater detail. A user need only read those parts which satisfy his particular need. A casual user who wants only simple retrieval operations using system defaults, has to read only a few pages. And the prompting facility can be used with only a paragraph or so of instruction.

Another user problem that must be treated is the separation of novices and experts. As is often the case, conversational systems are handled by users with widely varying degrees of expertness. The system should neither hamper the expert with excessive verbosity nor hinder the novice with obscure and terse responses. Some systems compromise and use a "middle of the road" approach, but this satisfies no one. Other systems have multiple sets of dialogue scripts. A user is classified as having a particular level of proficiency and he receives the dialogue appropriate to that level. But this too can lead to problems. In any large facility such as an information retrieval system, it is entirely possible for a user to be very proficient in some but not all areas of the system. Classifying him strictly as a novice or expert is wrong in both cases. To solve this problem, the proposed system uses an implicit rather than explicit separation of novice and expert. This is accomplished by allowing access to options only when the user asks for them. Thus the more the user knows about the system, the more facilities he has at his disposal. The novice is thereby protected from options which he does not understand. Tutorials are also presented only on request. Because of this only a single set of tutorials is needed and they can be reasonably long and clear. The expert user who does not ask for a tutorial need never see any and thus is not hindered by them. The only manifestation of the novice facilities that an expert must see is the short question:

Do you need help in using the system?

This appears immediately after signon. Even this can be eliminated by allowing a user status file to be stored between system uses. Upon signing on, the user's status file is read and appropriate parameters, including his negative answer to the above question, are set.

A few other characteristics of the proposed system also help in the proper handling of both novice and expert users. These are the multi-step processing technique and the ability to compound commands on a single line. An expert, for example, can put several system commands into a single input thus saving time and effort. The same commands may also be split on a number of lines for greater clarity. This and the multi-step process are discussed in greater detail in section 4.

One final goal of the proposed system is the presentation of useful tutorials. These messages must be easily available so that even the most confused user can get help. One simple method is to use a single question mark "?" as the tutorial request. The tutorials must reflect the specific place in the dialogue where they are called. In addition, they must take into consideration the commands and options that the user has already specified. Tutorials are also useful in treating errors. When an erroneous input is detected, the system automatically produces a tutorial appropriate to the place where the error occurs. The incorrect input is an implicit indication that the user needs help and thus the tutorial is appropriate at that point.

The design considerations presented in this section are basically nontechnical. They stem from an effort to satisfy within practical limits the basic conversational needs of the largest possible user population. The next section presents a discussion of the actual implementation of such a system.

4. Implementation of the Conversational System

This section discusses the implementation of the conversational system. The major obstacle in the process is the fact that the Cornell University Computing Center has at present, no facilities for user implementation of on-line systems. The programs thus must all be run in the conventional manner with batched input. This poses no real problem in the design and operation of the system except in the area of testing it on real users. But even this can be circumvented with adequate simulation.

A) Capabilities

The conversational system is designed to perform SMART-like information retrieval operations. The capabilities built into the present system include specification of a correlation coefficient, search strategy and collection to be used. The first two of these are provided with default values that are used if nothing is explicitly specified by the user. There is provision for submitting a query containing a number of data base entry point references (subject, date, journal, and author). A search can be initiated and the user can request to see any number of retrieved documents. In addition to these information retrieval operations, the user has available some commands to the conversational system itself. These include requesting a tutorial, asking to be guided through a retrieval operation, and signing on or off. A few other information retrieval operations, most notably relevance feedback, are deliberately omitted, since the system is designed to test the conversational and natural language capabilities, and not to retest the information retrieval techniques. The set of capabilities is selected as typical of the inputs, outputs and internal processes required in a larger system. Also relevance feedback is not conductive to handling in natural language. While a user might introduce a natural language input which indicates his desire to perform relevance feedback, the actual submission of relevancy judgements is best handled in a fixed format. Relevance feedback and a few other capabilities would add little to the significance of system experimentation and hence are omitted.

B) Input Conventions

While it is the aim of this system to allow natural language input, there are a few places where the use of natural language is impractical. This is usually caused by the physical characteristics of the conversational system or information retrieval in general. One such instance is in setting off a query from other types of input. A query may deal with any subject area. For example it could ask for information about some aspect of a conversational system. It could thus be indistinguishable from a legal system command. For this reason, the user rather than the system, must perform the discrimination between queries and commands. This is accomplished by simply prefacing each query with "QUERY" or "Q". This adds little to user effort and eliminates what might be an impossible system task. Another area where fixed format is necessary is in search initiation. Unlike other operations in a conversational system which require only a few computer cycles, the search is relatively costly in computer time. It is therefore desirable to avoid uncalled for searches. Also, searches should not be initiated until the user is satisfied with his query and search specifications. For these reasons, searches are performed only upon an explicit signal ("GOSEARCH") from the user. A third fixed format input is the request for a tutorial. This is accomplished by typing a single question mark ("?"). This is done strictly for user convenience. In this way, even the most confused user can receive a message appropriate to his present dialogue position. Tutorials are also automatically generated when a user introduces an incorrect input. The final fixed form input is the SIGNON command. In an actual on-line implementation, it is quite possible that this command will be handled by a supervisor program which controls all on-line operations. Thus the natural language analysis facility may not be present to process this input. The remainder of the inputs may be posed in natural English.

C) The Structure of the Process

The structure of the conversational system may be viewed as a graph. The nodes represent user decision points and the edges represent possible alternatives and system actions. As the user progresses through his dialogue, he moves from node to node in the graph. The action is much like that of a finite automaton. At every point in the dialogue, the user is at some system node. The combination of this current node and the user's input at that point determine the action to be performed (analogous to the output of the automaton) and the node to which control is passed after the action is completed. This strategy allows the system to be throught of as a set of modular units. Each unit corresponds to a node and each has associated with it the subset of inputs that are legal at that point, as well as the associated actions. The input processing is thus greatly simplified since at each node the system need only test for those inputs that are legal. All other inputs are illegal even though they might be acceptable at some other point in the dialogue. The modular approach also facilitates some degree of disambiguation. Some inputs are ambiguous when considered with respect to the total set of system inputs. However, many become unambiguous within the context of a single node. The simplest example is the tutorial request ("?"). The question mark by itself is not enough to determine which of the many tutorials is desired. But the combination of the question mark and the current node performs the disambiguation and the proper message is presented.

D) Template Analysis in the Conversational System

There are two main jobs to be performed in a natural language conversational system. The first is the natural language analysis required to transform the input to a machine-usable form. The second job is bookkeeping. The system must keep track of the user's present position in the dialogue, the legal inputs as well as the successor node associated with each input. It seems relatively clear that the template analysis process introduced by Weiss [31] is sufficient to handle the natural language analysis task. The expected input consists of queries and system commands coming from some sort of on-line terminal. They thus conform exactly to the user restricted input for which template analysis is designed. While more complex systems would produce a more rigorous analysis of the input, template analysis can provide all the information that is needed from the input and at a considerable saving in time over other methods. Thus template analysis appears to be the ideal natural language anlaysis technique for this application.

Upon first analysis the bookkeeping task seems outside the realm of template analysis. But actually, the most efficient way in which to implement this task is to imbed it within the template analysis structure. This is done as follows. Each template is applicable to only one node, which is called its host node. This is indicated by appending the host node number to the template concept numbers. Since template concept numbers range from 11 to 999, this appending can be accomplished by adding the desired node number times 1000 to the concept number. Each template contains a set of concept numbers, a key word, and a link to an action routine that is to be executed if that template is matched. Some additional information must be added for the conversational application. Each template must contain a next node: immediate (NNI) number which tells the node to which control is to be transferred immediately after execution of the associated template action. It is sometimes useful to defer transferring to a new node until all possible executions of the template action have been performed. For example in cases where a number of similar pieces of information must be picked up from one input. In this case, NNI refers to the host node. A second value, the next node: final (NNF) then indicates the node to which control is transferred after all actions at the current node are complete. In the examples in Fig. 2 below, template A and B are both applicable only in node 5, and both match the same input substring. After matching, however, template A calls action routine 51, and control is then immediately transferred to node 2. Template B causes action 55 to be performed and control remains at node 5. Finally, after all possible node 5 matches have been processed, control passes to node 3. In cases such as A where NNI causes a transfer to a node other than its own, the NNF value is ignored.

	NNI	NNF	ACTION	TEMPLATE CONCEPTS
A	2	-	51	5011, 5012, 5013
B	5	3	55	5011, 5012, 5013

Sample Conversational Templates

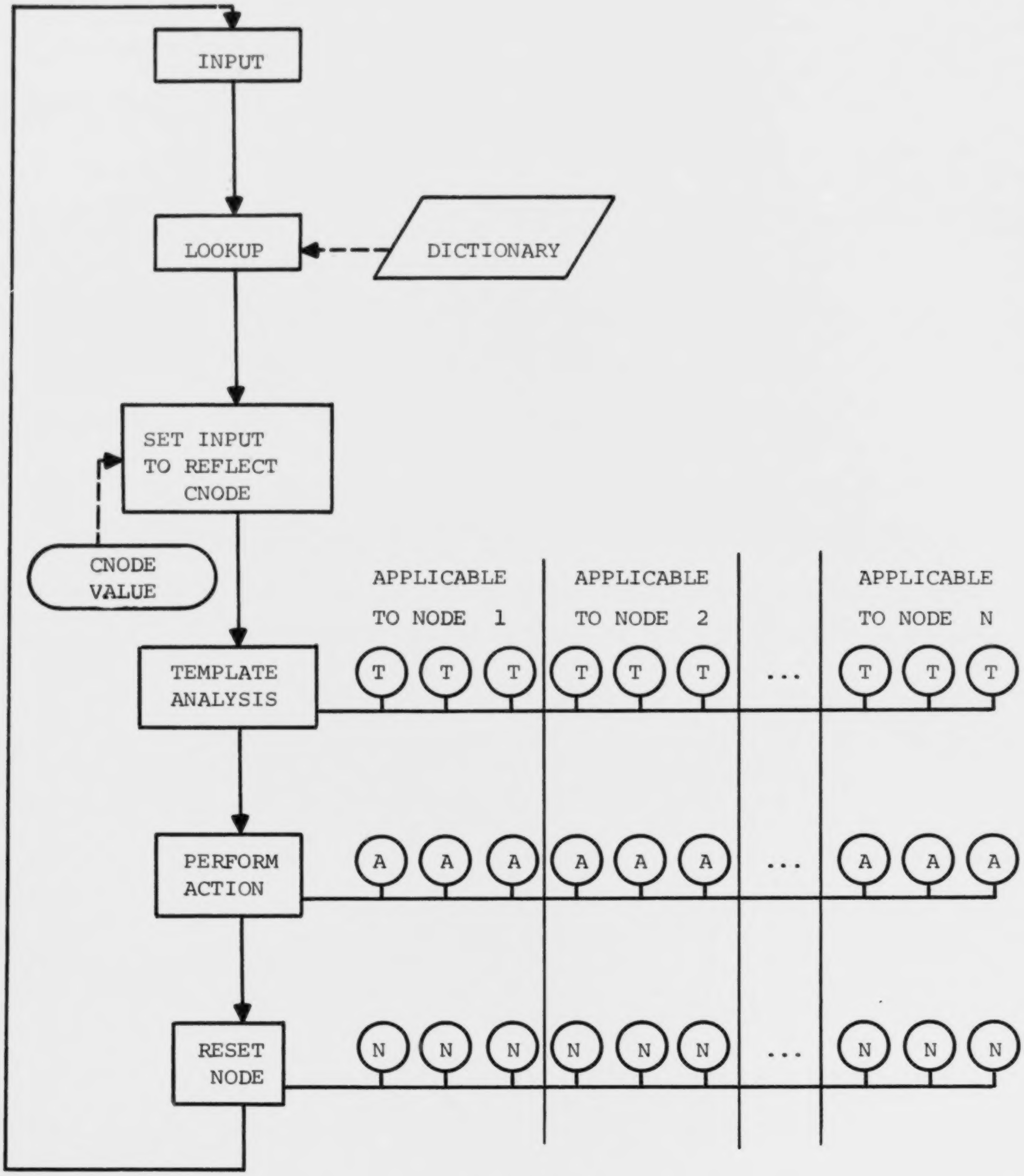
Fig. 2

In order to match the proper templates, the input must be made to reflect the current node (CNODE) in the dialogue. Upon reading an input, the current node times 1000 is added onto each input concept. Also, after every node change, the old node number is stripped off the input and the new node times 1000 added on. Thus the input reflects the current node in exactly the same way in which the templates reflect their host nodes and hence proper matching occurs. In this way the template process itself keeps track of the current node, the legal inputs for each node and the successor node function. This operation is summarized in the schematic in Fig. 3. An input is read and each word is assigned a numeric concept by a dictionary lookup. The input is then set to reflect the current node i . A scan is made of the entire template set in search of a match. However, only those templates whose host node is i have any chance of matching. If a match in this subset is found, the associated action is performed and the next node path is followed.

Fig. 4 indicates the node structure of the conversational system. Node 2 is the supervisor. After the initial signon phase, operations generally start and end in node 2. Most operations are two step processes. First, in node 2, the input is analyzed and the type of operation that it specifies is determined. Control then passes to the appropriate new node. Second, in this new node, the exact operation is determined and executed. Control is then returned to node 2. As an example consider the input

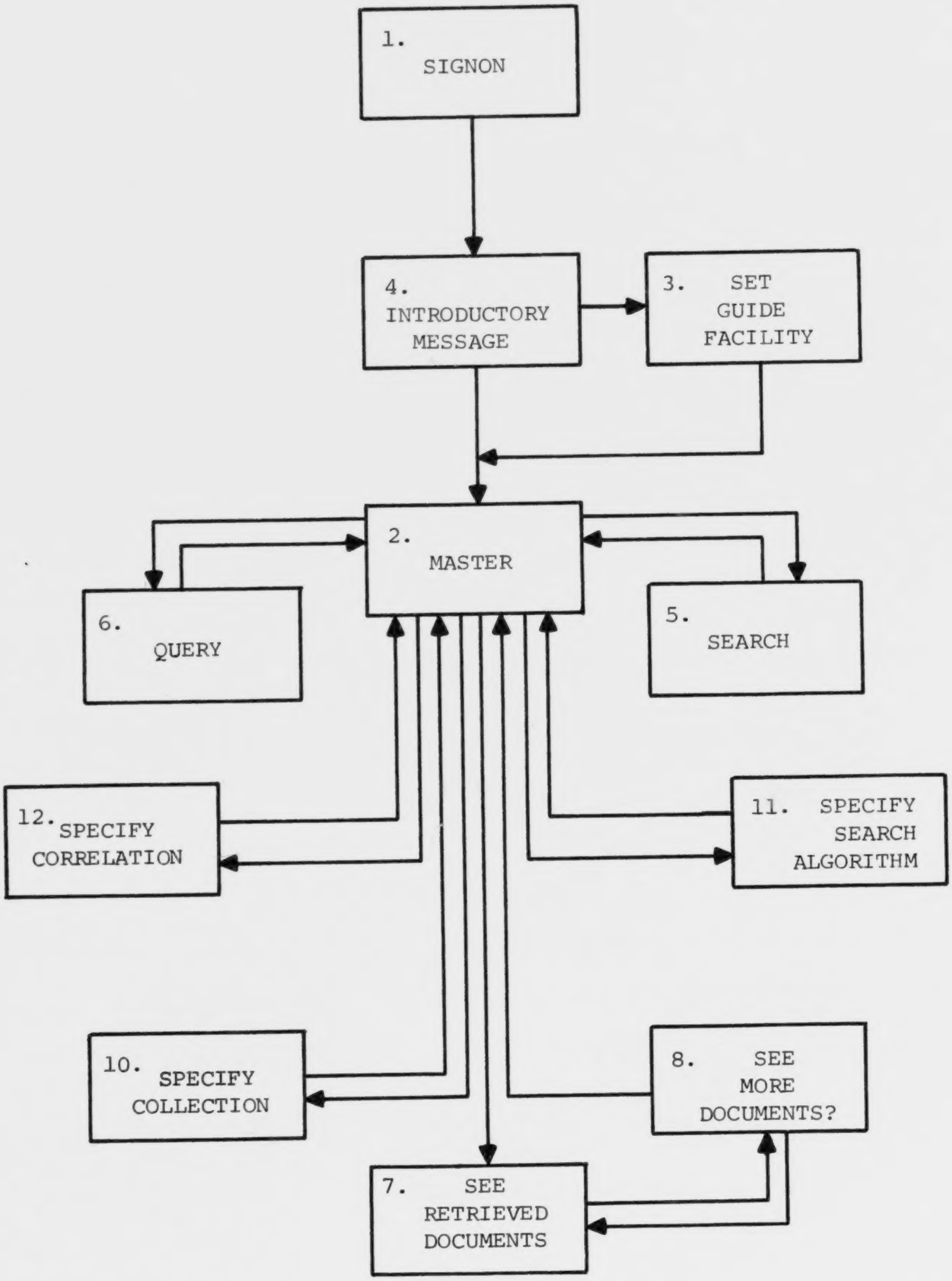
USE THE COSINE CORRELATION.

In node 2, it is determined that a correlation is to be specified and control passes to node 12. In node 12, the specific


INPUT
LOOKUP
DICTIONARY
SET INPUT TO REFLECT CNODE
CNODE VALUE
TEMPLATE ANALYSIS
PERFORM ACTION
RESET NODE
APPLICABLE TO NODE 1
APPLICABLE TO NODE 2
APPLICABLE TO NODE N
T
A
N
...
T=TEMPLATE
A=ACTION
N=NEXT NODE

Schematic of Conversational Operation

Figure 3


1. SIGNON
4. INTRODUCTORY MESSAGE
3. SET GUIDE FACILITY
2. MASTER
6. QUERY
5. SEARCH
12. SPECIFY CORRELATION
11. SPECIFY SEARCH ALGORITHM
10. SPECIFY COLLECTION
8. SEE MORE DOCUMENTS?
7. SEE RETRIEVED DOCUMENTS

Conversational Node Structure

Figure 4

correlation coefficient (i.e. cosine) is detected and noted. Control then goes back to node 2.

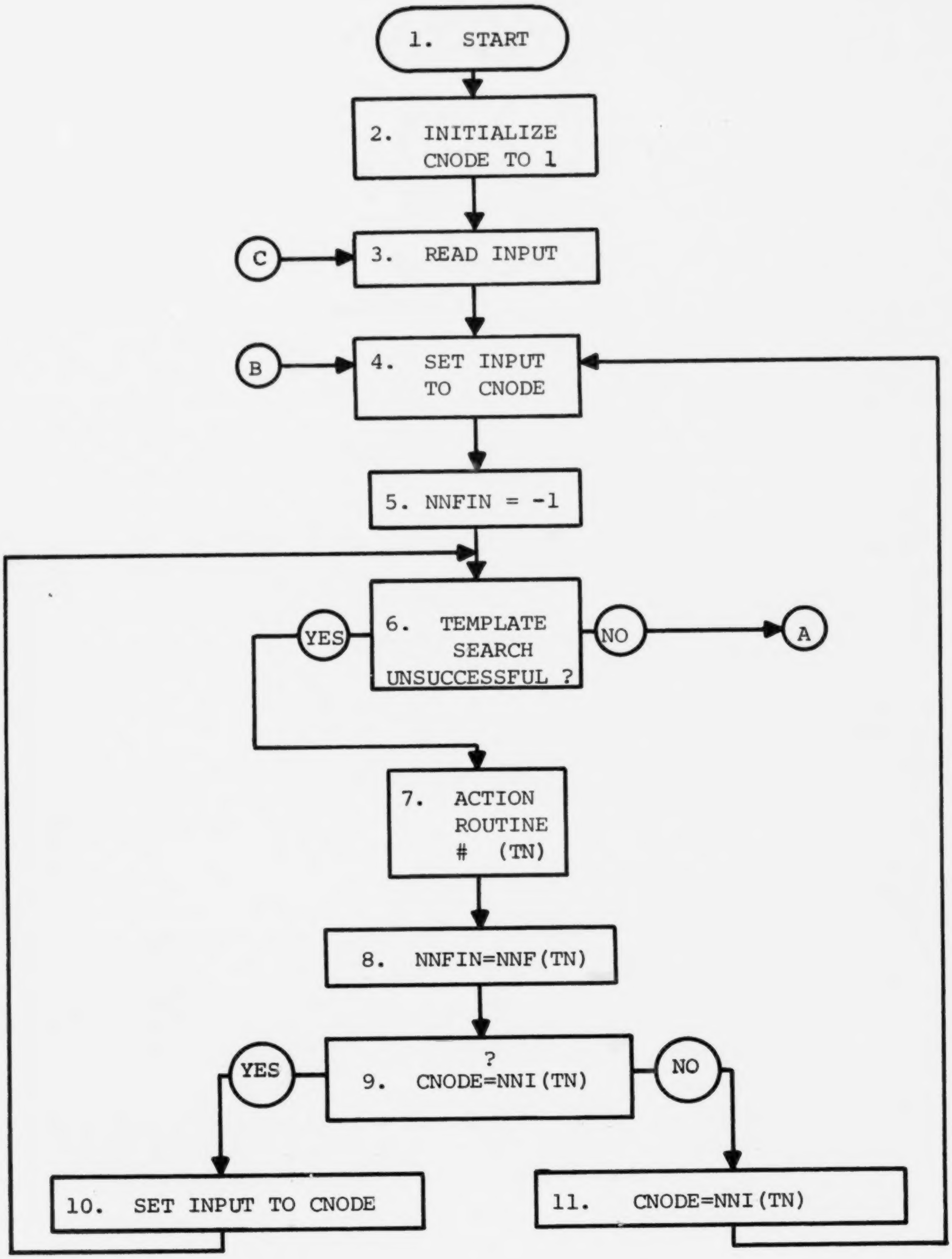
There is no necessity that the commands for two step operations appear on the same input line. For example, simply typing "CORRELATION" causes a transfer of control from node 2 to 12. The system then waits in 12 for further instructions. Strictly for the sake of convenience a special feature is used in cases like this. Whenever the system finds itself waiting in a node other than 2 it knows that an incomplete input has been entered. A special routine is therefore called to print a message appropriate to the current node. This aids the user in completing the input as is shown below. In this example and in all other samples of conversational scripts, user input is identified by a leading "U:".

```
U:  CORRELATION
SPECIFY A CORRELATION
U:  COSINE
```

Not only can inputs be spread out over several lines, several inputs can also be compounded onto a single line. For example

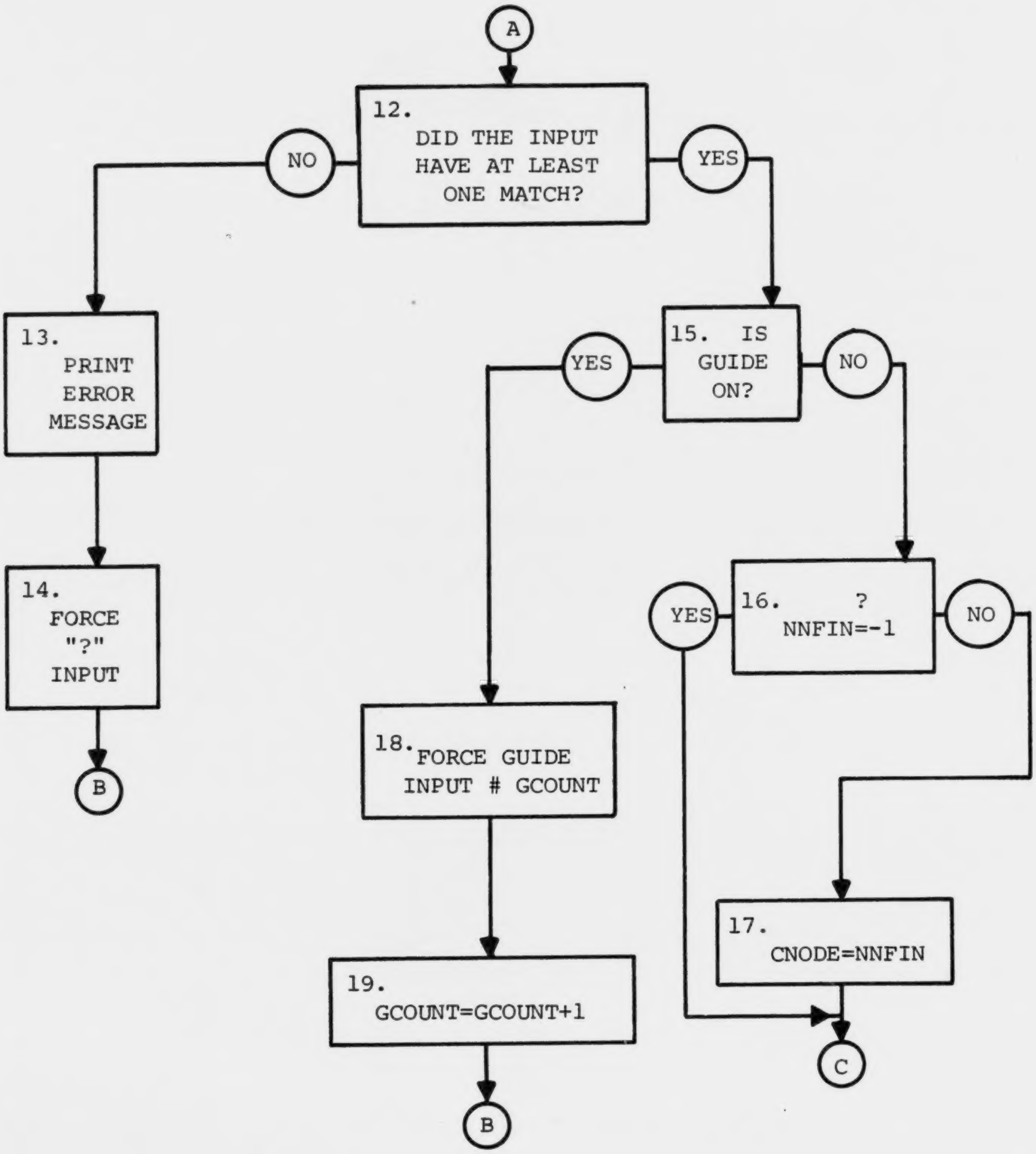
```
U:  PERFORM A FULL SEARCH ON THE PHYSICS COLLECTION
WITH THE COSINE CORRELATION.
```

As is seen in the detailed flow chart in Fig. 5, once an input is read, it is processed repeatedly until all valid template matches are exhausted. This results in an exit from box 6 via failure. Since this same exit is taken regardless of how many


1. START
2. INITIALIZE CNODE TO 1
C
3. READ INPUT
B
4. SET INPUT TO CNODE
5. NNFIN = -1
6. TEMPLATE SEARCH UNSUCCESSFUL ?
YES
NO
A
7. ACTION ROUTINE # (TN)
8. NNFIN=NNF(TN)
9. CNODE=NNI(TN) ?
YES
NO
10. SET INPUT TO CNODE
11. CNODE=NNI(TN)

Conversational Control Algorithm

Figure 5


A
12.
DID THE INPUT
HAVE AT LEAST
ONE MATCH?
NO
YES
13.
PRINT
ERROR
MESSAGE
14.
FORCE
"?"
INPUT
B
15. IS
GUIDE
ON?
YES
NO
16. ?
NNFIN=-1
YES
NO
18. FORCE GUIDE
INPUT # GCOUNT
19.
GCOUNT=GCOUNT+1
B
17.
CNODE=NNFIN
C

NOTE: NNFIN is the next node: final value. It is initialized to -1 before template matching begins. If no template matches are found, it will still be -1 at box 16. This indicates that control is to remain at the current node.

Conversational Control Algorithm

Figure 5 (Cond.)

or few, template matches occur in the input, a test must be made to see if at least one match occurs (box 12). If not, the input is not valid and a diagnostic must be presented to the user. The system prints a short general error message, erases the current input and replaces it by a question mark. Control is then passed back to the input analysis section. This results in the appropriate tutorial being shown to the user. This process of supplying diagnostics by allowing the system to force in a special input and then treating this as a normal user input is also used in the implementation of the guide facility which is discussed below.

E) The Guide Facility

In the original proposal for this system, a desire is expressed to provide a prompting facility to guide a novice user, step by step, through an actual retrieval operation. When a user signs onto this conversation system, he receives a brief introductory message:

> Do you need help in using this system?

If the user is familiar with the system he can simply answer NO and he sees no more of the prompting script. If his answer is YES, he receives a somewhat longer introduction to the system (See Fig. 6) and is then asked if he wishes to be guided through a retrieval operation. If not, the system operates normally and no prompting is given. If on the other hand, his answer to the second question is YES, the guide facility is turned on. The guide subroutine has a set of special strings of the form:

<general operation> ?

These include for example:

CORRELATION ?

SEARCH ?

QUERY ?

etc.

Each time the guide subroutine is called (see Fig. 5, boxes 18 and 19) it forces its ith string into the input area, increases i by one, and transfers control back to the input analyzer. These special inputs have the effect of performing the first half of a two step operation and then generating a tutorial. All the user has to do is respond in turn to each tutorial thus completing the second half of the two step process. When the guided retrieval process is finished, i is reset to one and the user is asked if he wants to be guided again.

F) Tutorials

There is a tutorial associated with each system node. When the user types a question mark, he is given the tutorial appropriate to his current node. The tutorials for all nodes except 2 provide instruction on the specific type of input expected. Unlike other nodes which have a very limited legal input set, almost all options are available from node 2. A different and more detailed form of tutorial message is necessitated in this case. The node 2 tutorial consists of two parts: the present status and the available options. The status report provides a summary of the specifications that the user has already made. The available options are presented as a list of tasks that are currently legal. Each option in the list has an identifying letter so that the user may pick it simply by typing the letter.

Fig. 6 shows some actual scripts produced by the conversational system with various levels of users ranging from novices who use the guide facility (Fig. 6A) to highly knowledgeable experts. The scripts include both correct and incorrect inputs as well as the various tutorial forms. The program which performs the conversation is written in G-level FORTRAN-IV. It consists of about 1300 FORTRAN statements and includes 35 subroutines and entry points. The program uses two output streams. One is used for diagnostic output. This is useful in debugging and in determining if the program's internal operations are working properly. The second stream is the conversational output which would be displayed on the user console. Timing statistics for the system are presented in section 5.

5. Experimentation

The experimentation tests the system with actual users and analyzes the results both with respect to system performance and user performance. The user population used represents a fairly broad spectrum of potential conversational system users with varying degrees of knowledge of information retrieval. The experiment for each user consists of two parts. First, the user is given a brief introduction to the system. Basically he is told that this is a conversational information retrieval system

```
U:  SIGNON
SMART ON-LINE...DO YOU NEED HELP IN USING THE SYSTEM?
U:  YES
THIS IS A NATURAL LANGUAGE CONVERSATIONAL
INFORMATION RETRIEVAL SYSTEM.  THE SYSTEM RETRIEVES
DOCUMENTS IN RESPONSE TO ENGLISH QUESTIONS.  YOU MUST SELECT
A DATA COLLECTION TO BE SEARCHED.  IN ADDITION YOU CAN
SELECT SEARCH STRATEGIES AND CORRELATION OR ALLOW SYSTEM
DEFAULTS.  IF YOU NEED HELP AT ANY TIME IN YOUR DIALOGUE,
TYPE "?" AND YOU WILL RECEIVE A TUTORIAL APPROPRIATE TO YOUR
PRESENT POSITION IN THE DIALOGUE.  FOR FULL INFORMATION
CONSULT THE MANUAL.

DO YOU WISH TO BE GUIDED THROUGH A RETRIEVAL PROCESS?
U:  YES
THE FOLLOWING COLLECTIONS ARE AVAILABLE FOR SEARCH
(SELECT BY NAME OR LETTER).
     A.  ENGINEERING
     B.  BIOLOGY
     C.  LITERATURE
     D.  CHEMISTRY
     E.  PHYSICS
U:  A
THE FOLLOWING CORRELATIONS ARE AVAILABLE.  CONSULT
MANUAL FOR FORMULAS.
     A.  COSINE (DEFAULT)
     B.  OVERLAP
U:  COSINE
THE FOLLOWING SEARCH STRATEGIES ARE AVAILABLE.
     A.  FULL SEARCH (CONSULT MANUAL BEFORE USING)
     B.  CLUSTERED (DEFAULT)
U:  CLUSTER
YOU MAY NOW ENTER YOUR QUESTION.  IT CAN BE IN NORMAL
ENGLISH AND MAY CONTAIN SPECIFICATIONS FOR DATES (YEAR),
JOURNAL, AND AUTHOR.
U:  INFORMATION ON ORDERING OF RECURSIVE FUNCTIONS BY THE
U:  DIFFICULTY OF COMPUTATION
YOU MAY NOW INITIATE A SEARCH BY TYPING GOSEARCH
U:  GOSEARCH
SEARCH UNDER WAY, EXPECT DELAY
SEARCH COMPLETE
```

CONVERSATIONAL DIALOGUE

Fig. 6A

```
INDICATE HOW MANY DOCUMENTS YOU WANT TO SEE.
IF NON TYPE "NONE".
U:  GIVE ME 5
RANK      1 DOC      91 CORR = 0.9896705
RANK      2 DOC      11 CORR = 0.9791623
RANK      3 DOC      42 CORR = 0.8563247
RANK      4 DOC      40 CORR = 0.8002142
RANK      5 DOC      45 CORR = 0.7656324
DO YOU WANT TO SEE MORE?
U:  YES 5 MORE
RANK      6 DOC      98 CORR = 0.6665342
RANK      7 DOC      25 CORR = 0.5542100
RANK      8 DOC      32 CORR = 0.4022313
RANK      9 DOC      89 CORR = 0.3987652
RANK     10 DOC      41 CORR = 0.3654201
DO YOU WANT TO SEE MORE?
U:  NO THANK YOU
SMART ON-LINE...DO YOU NEED HELP IN USING THE SYSTEM?
U:  NO
U:  ?
PRESENT STATUS:
     SEARCH ENGINEERING COLLECTION
     COSINE CORRELATION
     CLUSTERED SEARCH
     NO QUERY HAS BEEN SPECIFIED YET

AT THIS POINT YOU MAY:
     A.  SPECIFY A DOCUMENT COLLECTION
     B.  SPECIFY A SEARCH ALGORITHM
     C.  SPECIFY A CORRELATION COEFFICIENT
     D.  SUBMIT A QUERY
     F.  LOOK AT SOME RETRIEVED DOCUMENTS
     G.  START OVER
     H.  SIGNOFF
U:  H.  SIGNOFF
SIGNOFF ACCEPTED
```

CONVERSATIONAL DIALOGUE

Fig. 6A

(cont'd)

```
U:  SIGNON
SMART ON-LINE...DO YOU NEED HELP IN USING THE SYSTEM?
U:  NO
U:  YES
INPROPER INPUT
HELP FOLLOWS
PRESENT STATUS:
     SEARCH PHYSICS COLLECTION
     COSINE CORRELATION
     FULL SEARCH
     NO QUERY HAS BEEN SPECIFIED YET

AT THIS POINT YOU MAY
     A.  SPECIFY A DOCUMENT COLLECTION
     B.  SPECIFY A SEARCH ALGORITHM
     C.  SPECIFY A CORRELATION COEFFICIENT
     D.  SUBMIT A QUERY
     F.  LOOK AT SOME RETRIEVED DOCUMENTS
     G.  START OVER
     H.  SIGNOFF
U:  Q.  1970 LASER ARTICLES
QUERY SUMMARY:  DATE = 1970
U:  4
YOU CANNOT SEE DOCUMENTS UNTIL A SEARCH HAS BEEN PERFORMED
COMMAND IGNORED
U:  GOSEARCH
SEARCH UNDER WAY, EXPECT DELAY
SEARCH COMPLETE
U:  4
RANK      1 DOC      56 CORR = 0.9828174
RANK      2 DOC       5 CORR = 0.9620456
RANK      3 DOC      81 CORR = 0.8022564
RANK      4 DOC      21 CORR = 0.6553271
DO YOU WANT TO SEE MORE?
U:  NO
U:  SIGNOFF
SIGNOFF ACCEPTED
```

CONVERSATIONAL DIALOGUE

Fig. 6B

```
U:  SIGNON
SMART ON-LINE...DO YOU NEED HELP USING THE SYSTEM?
U:  NO
U:  ?
PRESENT STATUS:
     NO COLLECTION SPECIFIED YET
     COSINE CORRELATION (BY DEFAULT)
     CLUSTERED SEARCH (BY DEFAULT)
     NO QUERY HAS BEEN SPECIFIED YET

AT THIS POINT YOU MAY:
     A.  SPECIFY A DOCUMENT COLLECTION
     B.  SPECIFY A SEARCH ALGORITHM
     C.  SPECIFY A CORRELATION COEFFICIENT
     D.  SUBMIT A QUERY
     G.  START OVER
     H.  SIGNOFF

U:  CORRELATION
PLEASE SPECIFY A CORRELATION
U:  ?
THE FOLLOWING CORRELATIONS ARE AVAIABLE FOR SEARCH.
CONSULT MANUAL FOR FORMULAS.
     A.  COSINE (DEFAULT)
     B.  OVERLAP
PLEASE SPECIFY A CORRELATION
U:  B
U:  FULL SEARCH OF THE PHYSICS COLLECTION
U:  Q:  MU MESON PARTICLES
U:  GOSEARCH
SEARCH UNDERWAY, EXPECT DELAY
SEARCH COMPLETE
U:  I WANT 5 DOCUMENTS
RANK     1 DOC      53 CORR = 0.86547531
RANK     2 DOC      41 CORR = 0.7024357
RANK     3 DOC      49 CORR = 0.6654654
RANK     4 DOC       9 CORR = 0.5557575
RANK     5 DOC      17 CORR = 0.4302142
DO YOU WANT TO SEE MORE?
U:  NO
U:  SIGNOFF
SIGNOFF ACCEPTED
```

CONVERSATIONAL DIALOGUE

Fig. 6C

and that he must type "SIGNON" to begin. From then on, the user is on his own. The intent here is to see if the uninitiated user elects the guide option and if so, is the user successfully able to complete a retrieval operation using the guide facility? In the second experimental phase, the user tries to be more of an expert. Using information he has learned during the guided operation and some additional instruction, the user performs a second retrieval operation. This second operation is done without the aid of the guide facility. The sample scripts in Figure 6 are the actual results of these experiments with a few of the users. Results must be analyzed with respect to both system and user performance. For the most part, system performance can be measured objectively while user performance is more subjective.

A) System Performance

The basic measure of system performance is simply how many inputs are handled correctly out of the total number seen. This can be divided up since inputs arrive from several sources. Most inputs come directly from the user, but some are forced into the input area by the system itself. An input may be legal or illegal. Most illegal inputs are requests for options not accessible at the current node. If it is legal, a correct analysis is produced if the system performs the action intended by the user. For an illegal input, a correct analysis takes the form of noting the error and printing an appropriate message. Figure 7 shows for each input type, the total number of inputs, and the number analyzed correctly and incorrectly.

CONVERSATIONAL ANALYSIS

INPUT	TOTAL	# CORRECT	# INCORRECT	% CORRECT
LEGAL	295	293	2	99.3
ILLEGAL	10	8	2	80.0
FORCED	71	71	0	100.0
TOTAL	376	372	4	98.9

Summary of Conversational System Performance

Figure 7

In addition it shows the percent of correct analyses associated with this operation. These results indicate a very high level of performance for the system. Not only does it handle valid inputs successfully, but it is also able to detect invalid inputs and treat them properly. The total number of inputs shown in Figure 7 is actually greater than the total number of input lines. This is because several inputs may be compounded onto a single line.

B) User Performance

The measures of user performance are necessarily more subjective than those of system performance. However, these results can provide useful information into the overall validity of this type of approach to a conversational implementation.

For each user, at least two dialogues are conducted; one with the user having a minimum of system knowledge, and one where he has more instruction and previous experience. On the first try, every user responded properly to the initial system question and was able to turn on the guide facility. Then using the guide facility, all but one user was able to successfully complete a simple retrieval process. The one exception did not understand the use of the word "default". After this was explained, the operation progressed normally. In general, all users were able to respond properly to the guide questions. The only major problem occurred at the end of the guided dialogue where the process is recycled and started again. It was not obvious to the user at this point, how he could sign off. But most users knew enough to request a tutorial which then explicitly displayed the available options; SIGNOFF being one of them. An example of this situation appears in Figure 6A. A slight modification of the final guide process can rectify this.

Having been guided through retrieval operation supplies the user with a great deal of insight into the use of the system. Using this experience and a small amount of added instruction to fill in any areas not touched by the guide facility, the user next attempts a normal (unguided) dialogue. All of the users tested were able to conduct a reasonable dialogue without outside help. A few of the users who had previous information retrieval experience were able to perform a highly competent retrieval after only a single introductory guided process. Of course nearly all of the users became stuck at some point and had to request a tutorial. Of the 32 tutorial calls made by all users, all but one supplied the information necessary for the user to continue. In some cases where the user received the master status tutorial, the single message answered several of the user's questions. He was then able to continue by making several references back to the same message. The one situation in which the tutorial did not help occurred when a user requested a tutorial during a guide process. Since the guide facility operates by generating successive tutorial messages, the user's request resulted in a repeat of the previously printed message. Thus the tutorial presented no new information. The user, however, was able to extricate himself by requesting a default option. In all the dialogues there was no case in which a user was forced to stop because he became hopelessly lost.

At the conclusion of each user dialogue he is asked his opinion of the system. The reaction of nearly all the users was favorable. They found the system both simple to learn and use. The tutorial facility is very well received, especially the convention of printing the appropriate tutorial in response to an erroneous input. Most of the critical comments center around revision in the wording of the various messages. A few of these messages are felt to be insufficiently clear to a new user. One user suggested that tutorials not only explain their options but also provide some samples of appropriate valid inputs. This comment, however, appears to be based on user timidity more than anything else. Unlike others, this user did not fully appreciate the natural language capabilities of the system and was afraid of submitting an erroneous input. He therefore wanted the sample input as a highly structured quideline for his input. But because of the ability of the system to treat natural language, such guidelines are unnecessary.

The overall feeling of the user is that the system provides an easy to use yet sufficiently rigorous conversational information retrieval facility. In addition the control conversational dialogue can be performed at each user's particular level of competence.

C) Timing

No analysis of a potential on-line system is complete without saying something about processing time. The current conversational program is written in FORTRAN and contains a great deal of diagnostic processing and output, as well as other debugging aids. It might therefore be considered that the timing statistics for the program would be somewhat worse than could be achieved using more efficient production programming techniques. However, these results do give a general idea of the processing speed. The timing of each operation varies from about 50 to 150 milliseconds. The complete set of 376 operations is performed in 37.057 seconds or about 0.1 second per input operation. When considering an actual console user, a rather conservative estimate for the average time between inputs (that is the time between end of input signals) is 10 seconds. In practice this average is probably higher. Thus at the rate of 10 conversational operations per second, the current system could adequately support a network of 100 consoles and supply one second or better of response time. Even with the inefficient code and conservative estimates, this is clearly within practical limits.

6. Future Extensions

There are a number of areas for future study with respect to the conversational system. First is a user storage facility. With this capability a user could store various aspects of his dialogue, such as queries or retrieved documents, for future use. In addition, a user could store parameters which would be automatically set at sign-on time. This would eliminate the need to specify the parameters each time he used the system. In addition the system can keep various statistics about its own performance which are valuable in evaluating and improving the system.

Carrying the storage capability one step further, the conversational system could be equipped with a learning subsystem. A user could then specify his own notation along with more conventionally stated equivalents. The system would then learn the user's special requirements. In this way a user could tailor the conversational system to his exact needs and conventions. The learning process could also be used in the treatment of erroneous inputs. This is shown in the sample script below. The user erroneously requests a nonexistent "BOOL" correlation. The system notifies him of his error and requests clarification and whether the incorrect input should be learned. After answering affirmatively, the user may then use "overlap" or "bool" interchangeably.

```
U:  BOOL CORRELATION
INCORRECT CORRELATION, PLEASE CLARIFY AND
INDICATE IF INPUT SHOULD BE LEARNED.
```

U: YES, OVERLAP
UNDERSTOOD; BOOL = OVERLAP

Thus the learning process provides a way of meeting the particular needs of each individual user.

Some further work must also be done with respect to user terminals. Currently the most popular on-line communcation device is the teletype console. These are easy to use and relatively inexpensive. The most serious drawback is their slow output speed. A fairly simple tutorial may take 30 seconds or more to print. This can frustrate the user and needlessly tie up the terminal. Another type of terminal is based on a cathode ray tube (CRT). These permit almost instantaneous display of messages. In addition, part of the screen may be devoted to a prompting area. In this way the user always knows where he is in his dialogue and what options are currently available. Some CRT units have a light pen which allows selection of options by merely pointing the pen at the name of the desired option on the screen. However, there are several problems with CRT displays. First, the added hardware needed to drive a CRT makes them very expensive. Some work is being done by Bitzer [1] on the design of an inexpensive visual display unit which uses a plasma screen and slide projector. However, these are not yet commercially available. Alse the CRT produces no hard copy. A user might thus have to copy a long list of document numbers from the screen. The solution to this may be supplied by devices which contain both a visual and a hard copy facility. The user conducts his dialogue on the CRT. Whenever he receives something he wants saved, he indicates the appropriate subset of the script which is then printed. Such a device is currently being used experiemntally by the RIQS System at Northwestern University [13].

Another area for future study is the manner in which documents are displayed to the user. SMART and a number of other systems normally display only the document number. At best document numbers provide minimal information about the document's content. It might be better to store document titles or even abstracts on-line so that they may be seen by the user. This could be done best using a high capacity, low speed peripheral storage device. However, the expense of the dedicated storage device along with the prospect of having the terminal tied up printing abstracts, may make this technique uneconomical. Another possibility is to store document abstracts on microfiche. A set of microfiche and a reader would be supplied at each terminal station. The user would get a list of document numbers from the information retrieval system and then look them up off-line at the reader. Not only is the physical equipment for this cheaper than an on-line file, but also the fact that the scanning of abstracts is done off-line frees up the terminal for more useful work.

The fourth and probably most significant area for future development is the anlaysis of the conversational user. It is from this type of study that will come significant advances in tailoring systems to the actual needs of the system user.

7. Conclusion

Conversational information processing has many advantages over conventional batch methods. In this study it is shown that it is quite reasonable to conduct conversational information retrieval in a natural language framework. Furthermore the template analysis process proves to be a useful technique not only for handling the natural language input to a conversational system, but it can take care of the bookeeping as well. The conversational system implemented using these techniques is shown by actual user experimentation to provide an excellent communication medium between man and machine.

References

[1] Alpert, D., and D.L. Bitzer, Advances in Computer-based Education, Science, Vol. 167 (March 1970).

[2] Atherton, P., and R.R. Freeman, AUDACIOUS, AIP Report, AIP/UDC 7, April 1968.

[3] Berezner, S.C., H.C. Carney, J.A. Craig and C.R. Longyear, DEACON: Direct English Access anc Control, General Electric Co., Proceedings FJEC, Santa Barbara, California, 1966.

[4] Bergman, S., W. Franks, E. Rubinoff and M. Rubinoff, Experimental Evaluation of Information Retrieval through a Teletypewriter, CACM, Vol. 11, No. 9 (September 1968).

[5] Bobrow, D.G., Natural Language Input for a Computer Problemsolving System, in Semantic Information Processing, M. Minsky, Ed., MIT Press, Cambridge, Mass., 1968.

[6] Bolt, R.H., Computer-assisted Socratic Instruction, in Conversational Computers, W.D. Orr, Ed., John Wiley and Sons, Inc., New York 1968.

[7] Curtice, R.M., and P.E. Jones, An Operational Interactive Retrieval System, Arthur D. Little, Inc., 1969.

[8] Dimsdale, B., and B.G. Lamson, A Natural Language Information Retrieval System, Proc. of IEEE, Vol. 54, No. 12 (December 1966).

[9] Halpern, M., Foundations of the Case for Natural Language Programming, IEEE Spectrum, Vol. 4, No. 3, March 1967.

[10] IBM Systems/360 Document Processing System, Applications Description, IBM, 1967.

[11] Ide, E.C., Relevance Feedback in an Automatic Document Retrieval System, Information Storage and Retrieval, Report No. ISR-15 to the National Science Foundation, Cornell University, 1968.

[12] Kellogg, C.H., A Natural Language Compiler for On-line Data Management, AFIPS Conference Proceedings, Vol. 33, Proc. AFIPS 1968 Fall Joint Computer Conf., Vol. 33, Thompson Book Co., Washington, D.C.

[13] Krulee G., and B. Mittman, Computer-based Information Systems for University Research and Teaching, Northwestern University, Evanston, Illinois, 1969.

References (contd.)

[14] Maceyak, J., A Question-answering Language for a SMART type Data Base, May 1968.

[15] Mathews, W.C., TIP Reference Manual, Technical Information Program, The Libraries, MIT, Cambridge, Mass., 1968.

[16] Meister, D., and D.J. Sullivan, Evaluation of User Reactions to a Prototype On-line Information Retrieval System (RECON), Appendix RECON User's manual, Report NASA-CR 918, Prepared by Bunker-Romo Corporation, Conoga Park, California.

[17] Moyne, J.A., PROTO-RELADES: A Restrictive Natural Language System, IBM, 1967.

[18] Moyne, J.A., A Progress Report on the Use of English in Information Retrieval, IBM Corp., Federal Systems Center, Gaithersburg, Maryland, June 1969.

[19] Moyne, J.A., Information Retrieval and Natural Language, IBM Corp., Federal Systems Center, Gaithersburg, Maryland, June 1969.

[20] Orr W.D., (Ed.), Conversational Computers, John Wiley and Sons, Inc., New York, 1968.

[21] Parker, E., SPIRES User Manual, Stanford Physics Information Retrieval System, Institute of Communications Research, Stanford University, Palo Alto, California.

[22] Reitman, W., R.B. Roberts, R.W. Sauvain, and D.D. Wheeler, AUTONOTE: A Personal Information Storage and Retrieval System, Mental Health Research Institute Communication #248 and Information Processing Working Paper #12, University of Michigan, Ann Arbor, Michigan, 1969.

[23] Rubinoff, M., S. Bergman, H. Cautin, and F. Rapp, Easy English, A Language for Information Retrieval Through a Remote Typewriter Console, CACM, Vol. 11, No. 10 (October 1968).

[24] Salton, G., Automatic Information Organization and Retrieval, McGraw Hill, New York 1968.

[25] Salton, G., Interactive Information Retrieval, (Unpublished).

[26] Silvern, L.C., CAI in an Expanding Universe of Educational Methodology, in Conversational Computers, W.D. Orr, Ed., John Wiley and Sons, Inc., New York 1968.

References (Contd.)

[27] Simmons, R.F., Synthex, in Conversational Computers, W.D. Orr, Ed., John Wiley and Sons, Inc., New York, 1968.

[28] Simmons, R.F., Natural Language Question-answer Systems: 1969, CACM, Vol. 13, No. 1 (January 1969).

[29] Summit, R.F., DIALOG II Users Manual, Information Science Electronic Science Lab., Lockheed Palo Alto Research Lab., Lockheed Missiles and Space Co.

[30] Thompson, F.B., DEACON Type Query Systems, in Conversational Computers, W.D. Orr, Ed., John Wiley and Sons, Inc., New York, 1968.

[31] Weiss, S.F., A Template Approach to Natural Language Analysis for Information Retrieval, Ph.D. Thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1970.

[32] Weiss, S.F., Template Analysis and its Application to Natural Language Processing, Information Storage and Retrieval, Report No. ISR-16 to the National Science Foundation, Cornell University, 1969.

[33] Weizenbaum, J., Contextual Understanding by Computers, CACM, Vol. 10, No. 8 (August 1967).

[34] Weizenbaum, J., ELIZA — A Computer Program for the Study of Natural Language Communications Between Man and Machine, CACM, Vol. 9, No. 1 (January 1969).

[35] Williamson, R., A Prototype Document Retrieval System. (Unpublished).